Tutorial
Trend surfaces in R by Ordinary and Generalized Least Squares

D G Rossiter
Nanjing Normal University, Department of Geographic Sciences

November 17, 2017

Contents

Version 1.1 Copyright © 2017 D G Rossiter 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 (d.g.rossiter@cornell.edu).

1 Introduction

This exercise shows how to compute trend surfaces using the **R environment for statistical computing** [4, 8].

A *trend surface* is a map of some continuous variable, computed as a function of the coördinates. This corresponds to the concept of a *geographic trend*, where the variable changes its value along a geographic gradient. This can be a *linear* trend, i.e., the variable increases or decreases a fixed amount for each unit change in the coördinates in some direction. This is called a *first-order* trend surface. It can also be a *polynomial* trend, i.e., a linear model of some polynomials of the coördinates, for example, a quadratic, which is called a *second-order* trend surface.

> **Note:** It is possible to use a *non-linear* function of the coördinates, but we will not explore that in this exercise.

1.1 Example dataset

We use an example dataset that is well-suited to illustrate the concepts of trend surface: a set of observations on the elevation above mean sea level of the top of an aquifer in western Kansas, USA measured in 161 wells.

> **Note:** This aquifer is in Miocene–Pliocene sedimentary rocks, the Ogalalla formation, and is an important source of irrigation water, especially for centre-pivot irrigation systems.

This dataset is used as an example in the well-known geology statistics text of Davis [2, pp. 435-438][1]. The practical task is to map the elevation of the top of the aquifer over the study area.

Q1 : *What is the purpose of producing a map of the the elevation of the top of the aquifer over the study area? In other words, who would use the map and for what purpose?* *Jump to A1* •

> **Note:** More information on the aquifer monitoring network from which this dataset is taken is available at the Kansas Geological Survey[2], for example Olea and Davis [6, 7]. The water-level logs are also available on-line[3].

Figure 1 is taken from the original report [6]. It shows the location of wells, the boundary of the aquifer, and the well IDs. The example dataset uses a small portion of this, in the SE corner of the study area[4]. Figure 2 is a Google Earth view of part of the study area, with the location of several of the wells as placemarks.

[1] The datasets for this book are available at http://www.kgs.ku.edu/Mathgeo/Books/Stat/index.html
[2] http://www.kgs.ku.edu
[3] http://www.kgs.ku.edu/Magellan/WaterLevels/
[4] portions of Pratt, Kingman, Stafford and Reno counties

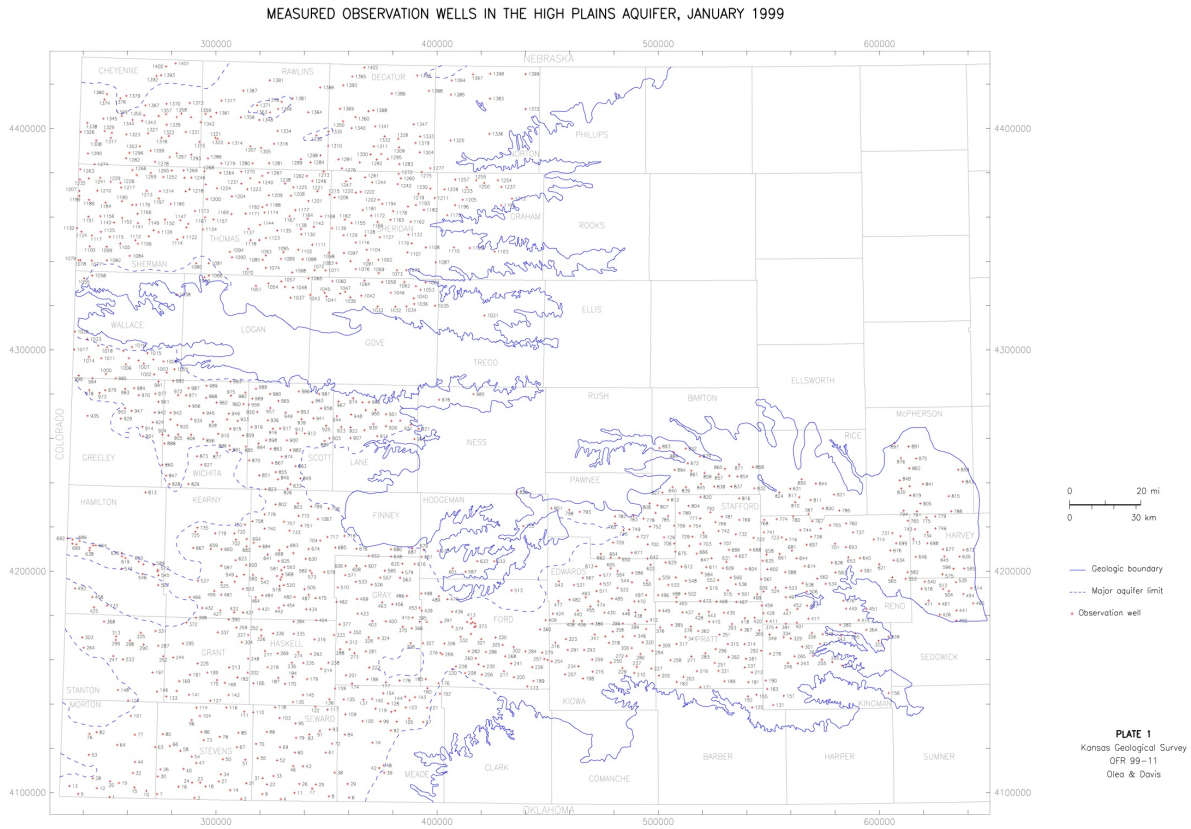


Figure 1: Location of aquifer monitoring wells, SE Kansas (USA). Source: [6], plate 1

2 Preparing for the exercise

The easiest way to complete these exercises is:

1. Start RStudio;
2. Load the code for this exercise, `TrendSurface_ex1.R`, into R Studio, using the File | Open ... menu command;
3. Make sure the **working directory** in R console is where you've downloaded the sample datafile `AQUIFER.TXT` (use the Tools | Set Working Directory ... menu item);
4. Pass the code step-by-step from RStudio to the R console using the "Run" toolbar button or the Code | Run lines ... menu item.

2.1 Loading R packages

Task 1 : Load the `sp` "spatial data structures", the `gstat` "geostatistics", and the `lattice` "Trellis graphics" packages. •

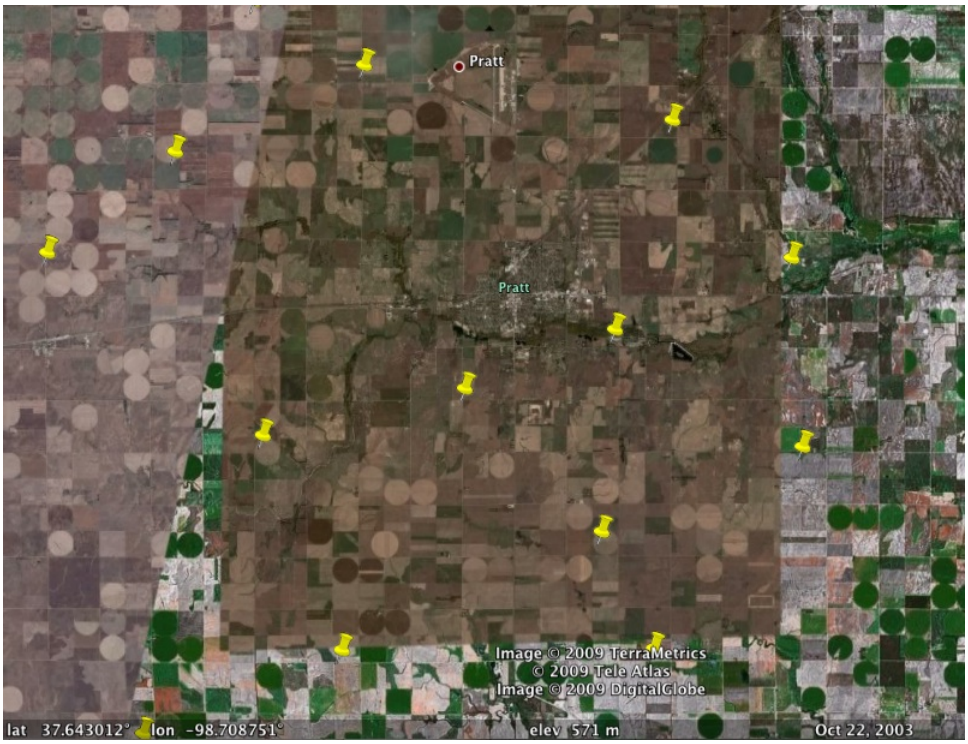


Figure 2: Google Earth view of part of the study area, with the location of several of the wells as placemarks

Note: You can also load this via checkboxes in the RStudio "Packages" pane.

The `require` or `library` functions are used to load R packages.

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

2.2 Loading and adjusting the dataset

Task 2 : Change R's *working directory* to where you have downloaded the text file `AQUIFER.TXT`. •

You can do this with the RStudio menu command Tools | Change directory. . . , or with the `setwd` function.

Task 3 : Examine the contents of file `AQUIFER.TXT`. •

You can view this file from within RStudio, by opening it from the Files pane.

The first few lines look like this:

```
UTM easting UTM northing Water Table, ft.
569464.5 4172114.75 1627.66
573151.25 4167192.75 1588.83
559973.94 4169585 1675.72
553514.44 4174584.5 1689.52
```

The field names are self-explanatory. The UTM zone is 14N (see Davis [2, Fig. 5-100 caption]) and the coördinates are meters. The aquifer elevation is in US feet[5] above mean sea level according to an unspecified vertical datum (probably NAVD 88).

Task 4 : Read text file `AQUIFER.TXT` into an R data frame, rename the columns to shorter names, and examine its structure. •

The `read.table` function can read many kinds of tabular data. It has many arguments, to adjust to different text formats. See the R Data Import/Export Manual [9] for details. By default the *data fields* in the text file are assumed to be separated by *white space* (tabs, spaces), as is the case here. Another optional argument is `skip`; we use it here because the header line of `AQUIFER.TXT` has more spaces than the other lines, so if we try to use the header for the variable names, R thinks the other lines are incomplete. One solution would be to place quotes around the variable names, or rename the variables, in the text file. What we do here is skip the first line and assign variable names ourselves in R.

We name the R data frame `aq`:

```
> aq <- read.table("AQUIFER.TXT", skip = 1)

> str(aq)

'data.frame':        161 obs. of  3 variables:
 $ V1: num  569464 573151 559974 553514 550350 ...
 $ V2: num  4172115 4167193 4169585 4174584 4171337 ...
 $ V3: num  1628 1589 1676 1690 1691 ...

> names(aq) <- c("UTM.E", "UTM.N", "z")
> str(aq)

'data.frame':        161 obs. of  3 variables:
 $ UTM.E: num  569464 573151 559974 553514 550350 ...
 $ UTM.N: num  4172115 4167193 4169585 4174584 4171337 ...
 $ z    : num  1628 1589 1676 1690 1691 ...
```

However, this dataset can be manipulated to make it more suitable for analysis. First, the elevation should be converted to meters, to conform to international standards.

[5] 1 foot = 0.3048 m exactly

Task 5 : Convert the elevation in feet above sea level to elevation in meters above sea level (m.a.s.l.), and add it as a new field in the dataframe. •

```
> ft.to.m <- 0.3048
> aq$zm <- aq$z * ft.to.m
```

Second, the E and N coördinates give the location in UTM zone is 14N, but for numerical stability it's useful to reduce these to local coördinates, with the $(0, 0)$ point in the middle of the range, and because the numbers are large, convert to km. This will make the equations easier to read.

Task 6 : Subtract the median E and N coördinates of the dataset from the UTM 14N E and N coördinates, convert these from m to km, and add these as new fields to the dataframe. •

The `median` function computes the median of a vector.

```
> aq$e <- (aq$UTM.E - median(aq$UTM.E))/1000
> aq$n <- (aq$UTM.N - median(aq$UTM.N))/1000
```

3 Exploratory spatial analysis

Task 7 : Summarize the dataset. •

```
> summary(aq)
```

```
     UTM.E            UTM.N              z              zm       
 Min.   :500361   Min.   :4150248   Min.   :1560.0   Min.   :475.50  
 1st Qu.:518465   1st Qu.:4176120   1st Qu.:1721.2   1st Qu.:524.61  
 Median :533366   Median :4197238   Median :1813.6   Median :552.79  
 Mean   :535668   Mean   :4198439   Mean   :1807.8   Mean   :551.01  
 3rd Qu.:553569   3rd Qu.:4220405   3rd Qu.:1901.0   3rd Qu.:579.42  
 Max.   :574430   Max.   :4248312   Max.   :2044.8   Max.   :623.24  
       e                  n          
 Min.   :-33.0050   Min.   :-46.9903  
 1st Qu.:-14.9011   1st Qu.:-21.1180  
 Median :  0.0000   Median :  0.0000  
 Mean   :  2.3014   Mean   :  1.2008  
 3rd Qu.: 20.2023   3rd Qu.: 23.1665  
 Max.   : 41.0632   Max.   : 51.0740  
```

Q2 : *How many observations are there? What was recorded at each point?* *Jump to A2* •

Q3 : *What are the geographic limits of the study area? What is its area, in km²?* *Jump to A3* •

The `range` function computes the range of numeric variable; the `diff` function computes the difference between two numeric values.

```
> range(aq$UTM.E)

[1] 500361.34 574429.56

> range(aq$UTM.N)

[1] 4150248.2 4248312.5

> diff(range(aq$UTM.E)) * diff(range(aq$UTM.N))/10^6

[1] 7263.4444
```

Task 8 : Find the location of this sample area in the large study area, shown in Fig. 1. •

Q4 : *What is the range of elevations in the sample set?* *Jump to A4* •

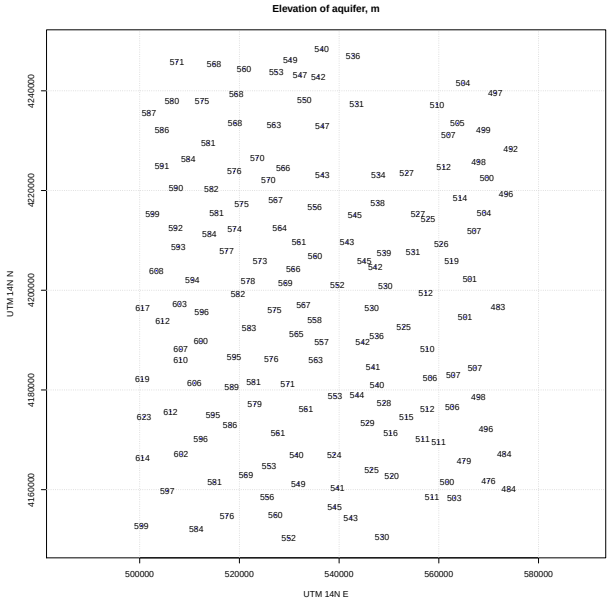
```
> range(aq$zm)

[1] 475.50324 623.24285

> diff(range(aq$zm))

[1] 147.73961
```

We now try three different **visualizations** of the distribution of the data values (i.e. aquifer elevations); these are known as **postplots**. To keep the geographic reference, we use the original UTM 14N coördinates.

Task 9 : Display a **text postplot** of the data values, showing the elevations, rounded to the nearest foot, as text labels centred at the observation point. •

We use the two coördinates as plot axes, so this looks like a map:

```
> plot(aq$UTM.N ~ aq$UTM.E, pch = 20, cex = 0.2, col = "blue",
+     asp = 1, xlab = "UTM 14N E", ylab = "UTM 14N N")
> grid()
> text(aq$UTM.E, aq$UTM.N, round(aq$zm), adj = c(0.5, 0.5))
> title("Elevation of aquifer, m")
```



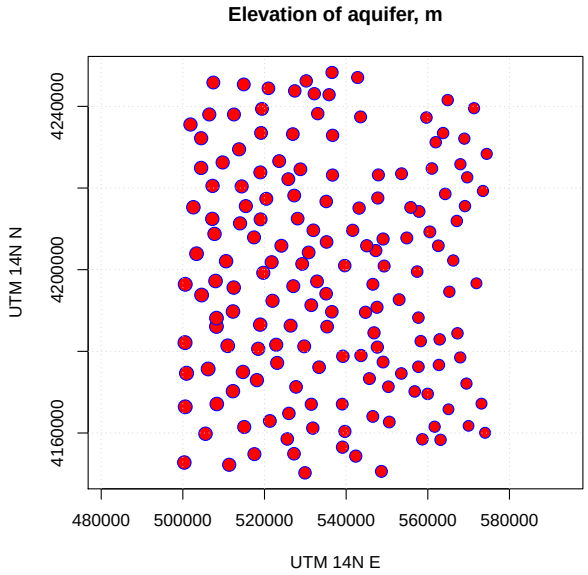
The aquifer elevation is clearly higher in the west (towards the Rocky Mountains about 650 km to the west, where it outcrops).

> **Note:** Parameter `cex` is an expansion factor; here we plot a very small blue dot and then add the data value at each point with the `text` method. The `adj` argument centres the text at the point. The `asp=1` argument makes the two axes have the same scale. This is necessary to get a true map when the study area is not square.

Another visualization is with the symbol size proportional to the the data value.

Task 10 : Display a graphical postplot of the data values, with size proportional to the data value. •

```
> plot(aq$UTM.N ~ aq$UTM.E, cex = 1.8 * aq$zm/max(aq$zm),
+     col = "blue", bg = "red", pch = 21, asp = 1, xlab = "UTM 14N E",
+     ylab = "UTM 14N N")
> grid()
> title("Elevation of aquifer, m")
```



Note: Print character (`pch`) 21 has both a symbol (`col`) and fill (`bg`) colour.

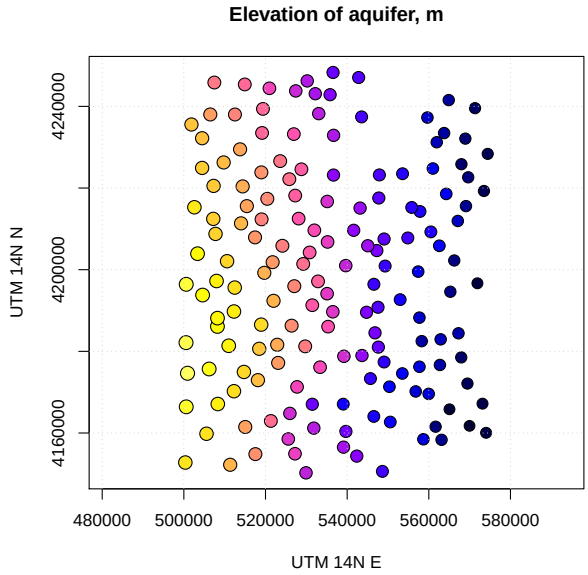
A final visualization combines both size and colour:

Task 11 : Display a graphical postplot of the data values, with size and colour proportional to the data value •

Notice the use of the `rank` function to give the rank order of the elevations; these are then used as indices into a vector of colours, created with the `bpy.colors` function, of the same length as the vector of elevation values.

The ~ formula operator show the functional relation between two variables; here it is the North coördinate for the y-axis, depending on the East coördinate for the x-axis.

```
> plot(aq$UTM.N ~ aq$UTM.E, pch=21,
+      xlab="UTM 14N E", ylab="UTM 14N N",
+      bg=bpy.colors(length(aq$zm))[rank(aq$zm)],
+      cex=1.8*aq$zm/max(aq$zm), asp=1)
> grid()
> title("Elevation of aquifer, m")
```



Q5 : *Describe the spatial pattern of the elevations. Do nearby points have similar values? Is there a trend across the whole area? Are there local exceptions to the trend?* *Jump to A5* •

Q6 : *Discuss the relative advantages of the three types of postplot.* *Jump to A6* •

4 Trend surface analysis by Ordinary Least Squares

The visualizations suggest a **trend surface**, i.e., the aquifer elevations as a linear model of the coördinates. This is a polynomial function of the coördinates to any degree (1st, 2nd, 3rd etc.), which is called the **order** of the surface. The higher the degree, the more the surface can match the points, but the degree should also be chosen to match a plausible process, in this case, the structure of the aquifer.

4.1 First-order trend surface

We begin with a **first-order** trend: a plane defined by the two coördinates and an intercept that sets the overall level, here the aquifer elevation.

Q7 : *What is the geological interpretation of a first-order trend surface of the aquifer?* *Jump to A7* •

A trend surface has the same form as a standard linear model, using the coördinates as regression predictors. The first-order trend surface model has the form:

$$z = \beta_0 + \beta_1 E + \beta_2 N + \varepsilon \quad (1)$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, i.e., independently and normally distributed. This assumption allows us to fit the trend surface with Ordinary Least Squares (OLS)

In the linear model, with any number of predictors, there is a $n \times p$ **design matrix** of predictor values usually written as $\mathbf{X}$, with one row per observation (data point), i.e., n rows, and one column per predictor, i.e., p columns. In the first-order trend surface case, it is a $n \times 3$ matrix with three columns: (1) a column of 1 representing the intercept, to center the response, (2) a column of predictor values e_i from the Easting, and (3) a column of predictor values n_i from the Northing. The predictand (response variable), here the aquifer elevation is a $n \times 1$ column vector $\mathbf{y}$, one row per observation. The coefficient vector β is a $p \times 1$ column vector, i.e., one row per predictor (here, 3). This multiplies the design matrix to produce the response:[6]

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \tag{2}$$

where ε is a $n \times 1$ column vector of **residuals**, also called **errors**, i.e., the lack of fit. We know the values in the predictor matrix $\mathbf{X}$ and the response vector $\mathbf{y}$ from our observations, so the task is to find the optimum values of the coefficients vector β. This can be found directly; see the Appendix A for the derivation. The OLS solution is:

$$\hat{\beta}_{\text{ols}} = (X^T X)^{-1} X^T \cdot y \tag{3}$$

where X is the design matrix.

The term "first-order" refers to the power to which each coördinate is raised; here it is the first power, so it's a first-order trend surface.

> **Note:** This assumption of uncorrelated residuals is in fact *not* true in this case; we prove this in §6 below. So the trend surface should in fact be fit not by OLS but by Generalized Least Squares (GLS), taking into account the spatial auto-correlation of the residuals. We pursue this further in §7.
>
> For this dataset with many observations well-spread in space, the result will be similar to the OLS estimate.

Task 12 : Fit a first–order trend surface (i.e. linear in the E and N coördinates) to the elevations. Summarize the model and evaluate its goodness-of-fit. •

The `lm` "linear model" function fits linear models.

```
> model.ts1 <- lm(zm ~ n + e, data = aq)
> summary(model.ts1)

Call:
lm(formula = zm ~ n + e, data = aq)

Residuals:
```

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

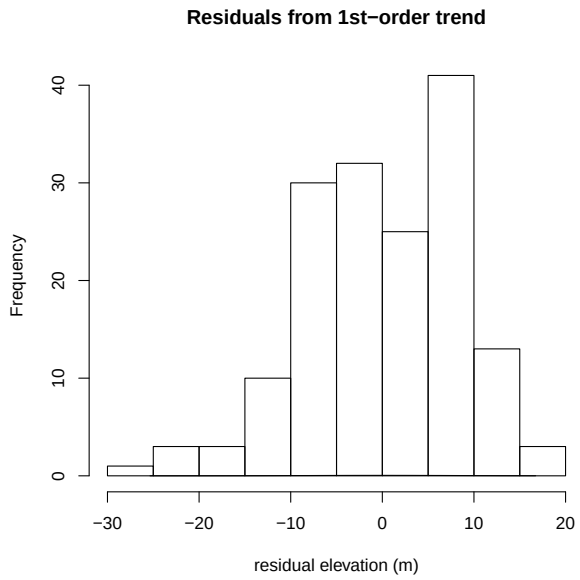
```
      Min        1Q    Median        3Q       Max
-25.35498  -5.82668   0.26742   7.10623  16.73489

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) 554.775093   0.684781 810.1496   <2e-16 ***
n            -0.033361   0.025281  -1.3196   0.1889
e            -1.617135   0.032014 -50.5132   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.6286 on 158 degrees of freedom
Multiple R-squared:  0.94169,        Adjusted R-squared:  0.94095
F-statistic: 1275.8 on 2 and 158 DF,  p-value: < 2.22e-16
```

Q8 : *What is the equation of the trend surface? How does elevation vary with the E and N coördinates? Is the relation statistically-significant? How much of the total variability does it explain? Are all the coefficients statistically-significant?* *Jump to A8* •

Task 13 : Summarize the **residuals** (lack of fit) from the trend surface both numerically and graphically, in feature space. Express this in terms of the median elevation. •

The `residuals` function extracts the residuals from a linear model object. The `hist` function displays a histogram of a numeric vector.

```
> res.ts1 <- residuals(model.ts1)
> summary(res.ts1)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-25.35498  -5.82668   0.26742   0.00000   7.10623  16.73489

> hist(res.ts1, main = "Residuals from 1st-order trend",
+     xlab = "residual elevation (m)")
> pts <- seq(min(res.ts1), max(res.ts1), length = 101)
> lines(pts, dnorm(pts, mean = mean(res.ts1), sd = sd(res.ts1)))
> curve(rnorm, xname = "res.ts1", )
> max(abs(res.ts1))/median(aq$zm) * 100

[1] 4.5867691
```



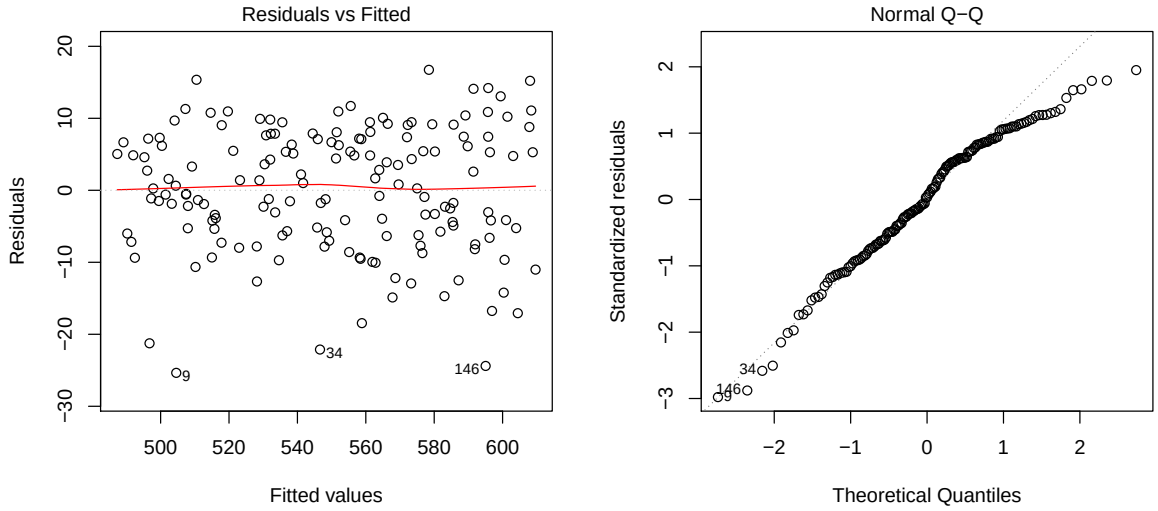
Q9 : *What is the range of residuals? How does this compare with the target variable? How are they distributed in feature space?* *Jump to A9* •

Task 14 : Show the diagnostic plots of the residuals •

The `plot` method applied to a linear model object produces some *diagnostic plots*. We will display the most important: (1) residuals vs. fitted values; (2) quantile-quantile ("QQ") plot of the standardized residuals.

The Q-Q plot shows (1) on the y-axis, the standardized residuals, (2) on the x-axis, the standardized residuals that *would be expected* if the residuals were from a normal distribution with the mean and standard deviation computed from the actual standardized residuals. (See §B for details on these residuals). These two should match exactly on 1:1 line.

```
> par(mfrow=c(1,2))
> plot(model.ts1, which=1:2)
> par(mfrow=c(1,1))
```



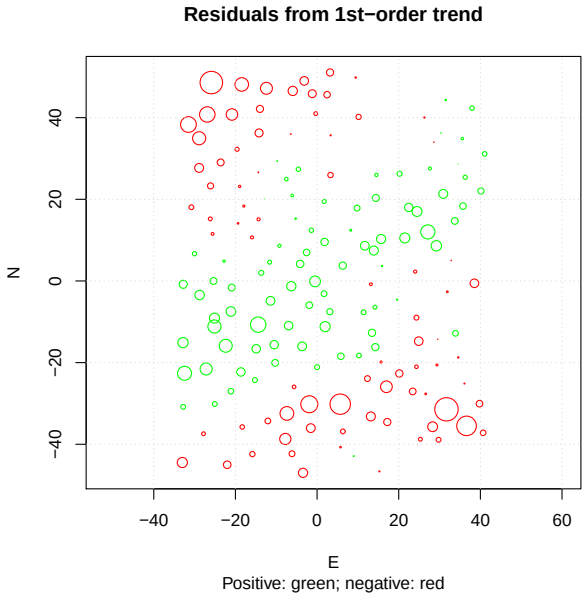
Q10 : *Does this model meet the feature-space requirements for a valid linear model?*

1. *No relation between the fitted values and the residuals;*
2. *Normally-distributed standardized residuals.*

Jump to A10 •

Task 15 : Display the residuals as a postplot. •

```
> plot(aq$n ~ aq$e, cex=3*abs(res.ts1)/max(abs(res.ts1)),
+      col=ifelse(res.ts1 > 0, "green", "red"),
+      xlab="E", ylab="N",
+      main="Residuals from 1st-order trend",
+      sub="Positive: green; negative: red", asp=1)
> grid()
```



Q11 : *Is there a spatial pattern to the residuals? Is there local spatial correlation without an overall pattern?* *Jump to A11* •

4.2 Second-order trend surface

We see from the pattern of residuals from the first-order surface that there is still structure, in particular clear bands of positive and negative residuals. These suggest that a higher-order trend surface might fit better.

Q12 : *What is the geological interpretation of a second-order trend surface of the aquifer?* *Jump to A12* •

Task 16 : Fit a second-order trend surface to the aquifer elevations. •

A full second-order surface uses the coördinates, their squares, and their cross-products.

$$z = \beta_0 + \beta_1 E + \beta_2 N + \beta_3 E^2 + \beta_4 N^2 + \beta_5 (E * N) + \varepsilon \tag{4}$$

```
> model.ts2 <- lm(zm ~ n + e + I(n^2) + I(e^2) + I(e *
+     n), data = aq)
> summary(model.ts2)

Call:
lm(formula = zm ~ n + e + I(n^2) + I(e^2) + I(e * n), data = aq)

Residuals:
      Min        1Q    Median        3Q       Max
-19.84694  -3.36558   0.82202   3.53842  14.80713
```

```
Coefficients:
               Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  5.6106e+02  7.9990e-01 701.4110 < 2.2e-16 ***
n           -1.6555e-02  1.6644e-02  -0.9946    0.3215
e           -1.6207e+00  2.2119e-02 -73.2739 < 2.2e-16 ***
I(n^2)      -7.4997e-03  6.4350e-04 -11.6546 < 2.2e-16 ***
I(e^2)      -1.6476e-03  1.0742e-03  -1.5338    0.1271
I(e * n)     6.6999e-03  7.7813e-04   8.6103 7.743e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.5985 on 155 degrees of freedom
Multiple R-squared:  0.97592,        Adjusted R-squared:  0.97514
F-statistic: 1256.3 on 5 and 155 DF,  p-value: < 2.22e-16
```

Note the use of the `I` "identity" function for the squares and cross-product; if this function is not used, `lm` interprets the `^` and `*` symbols as formula operators, rather than as their normal mathematical meanings.

Q13 : *How much of the variance does the second-order surface explain?* *Jump to A13* •

Task 17 : Compare the second-order model statistically with the first-order model. •

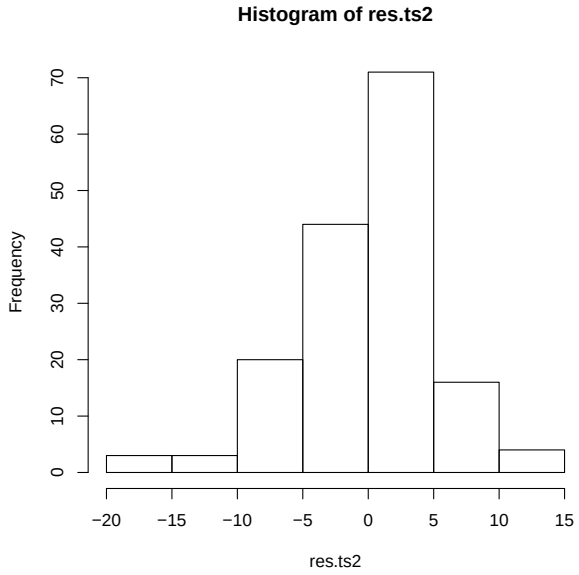
The `anova` "analysis of variance" method compares the residual sums-of-squares of two or more models and computes the probability that the more complicated model is not better than the less complicated model:

```
> anova(model.ts2, model.ts1)

Analysis of Variance Table

Model 1: zm ~ n + e + I(n^2) + I(e^2) + I(e * n)
Model 2: zm ~ n + e
  Res.Df      RSS Df Sum of Sq       F     Pr(>F)
1    155  4858.18
2    158 11763.49 -3  -6905.31 73.4379 < 2.22e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Q14 : *Is the second-order surface statistically superior to the first-order surface?* *Jump to A14* •

Task 18 : Summarize the **residuals** from the second-order trend surface both numerically and graphically, in feature space. Express this in terms of the median elevation. •

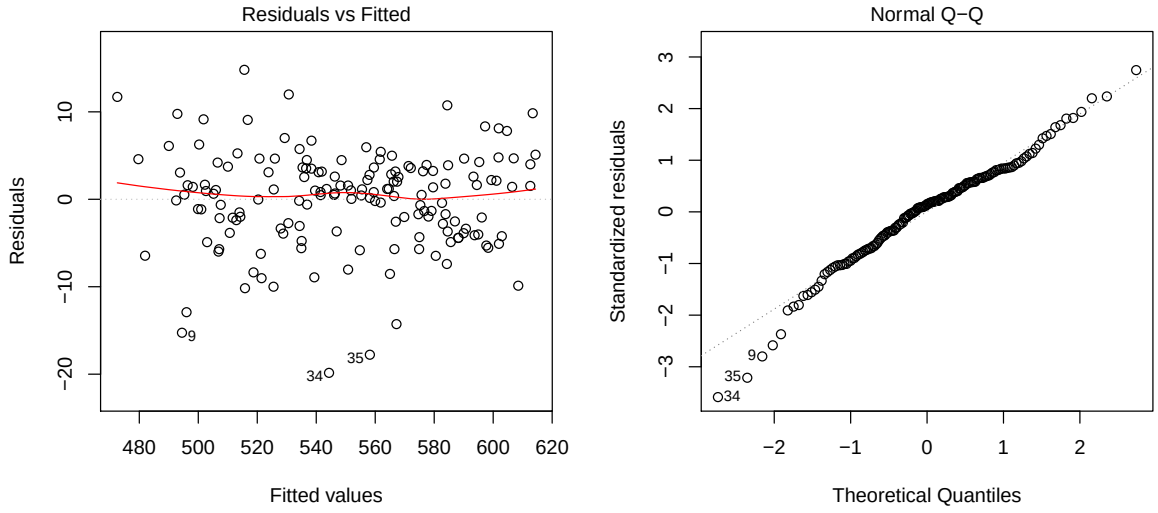
```
> res.ts2 <- residuals(model.ts2)
> summary(res.ts2)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-19.84694  -3.36558   0.82202   0.00000   3.53842  14.80713

> hist(res.ts2)
> max(abs(res.ts2))/median(aq$zm)

[1] 0.035903517
```



Q15 : *What is the range of residuals? How does this compare with the target variable? How are they distributed in feature space? How do these compare with the residuals from the first-order surface?* *Jump to A15* •

Task 19 : Show the diagnostic plots of the residuals, as for the first-order trend surface residuals. •

```
> par(mfrow=c(1,2))
> plot(model.ts2, which=1:2)
> par(mfrow=c(1,1))
```



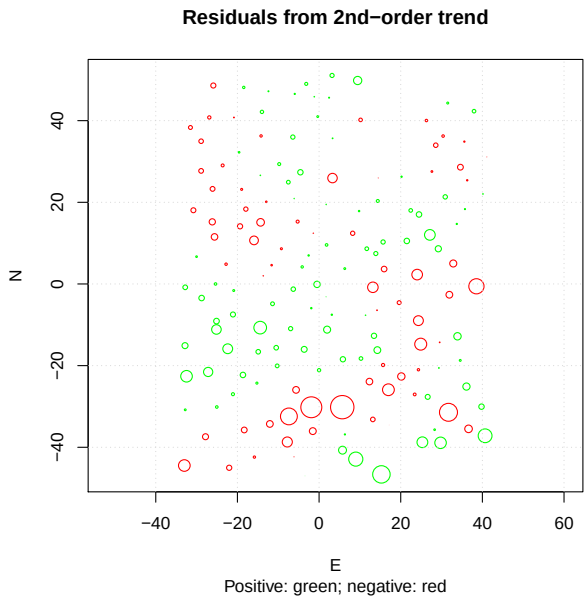
Q16 : *Does this model meet the feature-space requirements for a valid linear model? How do these diagnostics compare to those from the first-order surface?*

1. *No relation between the fitted values and the residuals;*
2. *Normally-distributed standardized residuals;*

Jump to A16 •

Task 20 : Display the residuals as a postplot; compare to the postplot from the first-order trend surface. •

```
> plot(aq$n ~ aq$e, cex=3*abs(res.ts2)/max(abs(res.ts2)),
+      col=ifelse(res.ts2 > 0, "green", "red"),
+      xlab="E", ylab="N",
+      main="Residuals from 2nd-order trend",
+      sub="Positive: green; negative: red", asp=1)
> grid()
```



Q17 : *Is there an overall pattern to the residuals? Is there local spatial correlation without an overall pattern? Does there seem to be any* anisotropy *(stronger spatial dependence in one direction than the orthogonal direction)?* *Jump to A17* •

Since this second-order trend surface is much better than the first-order trend surface, we will use it for subsequent modelling.

5 Trend surface prediction

This exercise uses the trend surface model of the previous section to **predict** over an **interpolation grid**.

5.1 Creating a prediction grid

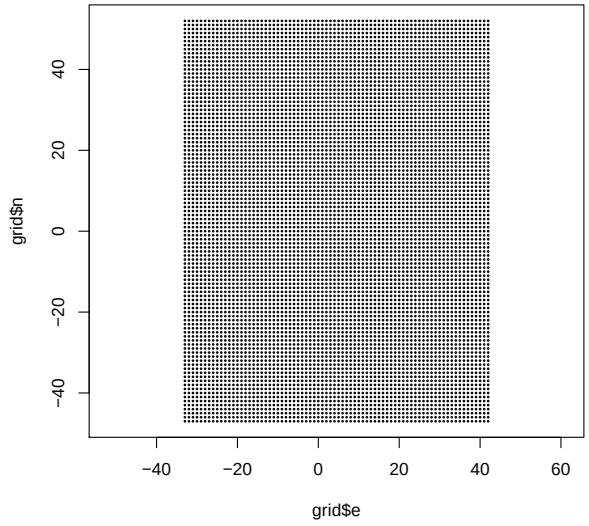
We first make a grid on which to predict.

Task 21 : Create a grid of equally-spaced (1 x 1 km) points across the study area, beginning with UTM (500 000E, 4150 000N) in the lower-left corner, as in Davis [2, Fig. 5-100, 5-101, 5-102], but adjusted for the reduced coördinates. •

The `seq` function creates a regular sequence of numbers; the `expand.grid` function makes a grid from two sequences.

```
> range(aq$e)

[1] -33.00497  41.06325

> range(aq$n)
```

```
[1] -46.99025  51.07400

> seq.e <- seq(-33, 42, by = 1)
> seq.n <- seq(-47, 52, by = 1)
> grid <- expand.grid(e = seq.e, n = seq.n)
> plot(grid$n ~ grid$e, cex = 0.2, asp = 1)
```



5.2 Mapping the trend surface

Task 22 : Interpolate the second-order trend surface onto this grid. Compute both the best fit and a 95% prediction interval for each point on the grid. •

The `predict.lm` function, applied to a linear model object, computes the predicted values at new locations, in this case the regular grid. The optional `interval` argument specifies that a prediction interval, as well as the best fits, should also be computed. The optional `level` argument specifies the $(1 - \alpha)$ probability, where α is the probability that, on repeated calculation from a similar sample, the true value at the point would not be included in the computed prediction interval.

```
> pred.ts2 <- predict.lm(model.ts2, newdata=grid,
+                        interval="prediction", level=0.95)
> summary(pred.ts2)

      fit              lwr              upr
 Min.   :461.07   Min.   :448.96   Min.   :473.17
 1st Qu.:516.41   1st Qu.:505.15   1st Qu.:527.72
 Median :547.31   Median :536.06   Median :558.56
 Mean   :546.68   Mean   :535.39   Mean   :557.97
 3rd Qu.:577.01   3rd Qu.:565.73   3rd Qu.:588.22
 Max.   :614.63   Max.   :603.23   Max.   :626.03
```

The `predict.lm` produces three fields in the resulting object: `fit` (the best fit value), `lwr` (the value at the lowest 2.5% limit) and `upr` (the value at the upper 2.5% limit).

The prediction interval is a range in which future observations are expected to fall, with a given probability specified by the analyst. It is based on the known observations and the regression model..

There are two sources of prediction error:

1. The uncertainty of fitting the best regression parameters from the available data;
2. The uncertainty in the prediction, even with perfect regression parameters, because of uncertainty in the process which is revealed by the regression, i.e., the inherent noise in the process.

The prediction interval is computed from the *prediction variance*, which is then assumed to represent the variance of a t-distribution.

The prediction variance $s^2_{Y_0}$ for predictand x_0 depends on the variance of the regression $s^2_{Y.x}$ but also on the distance of the predictor x_0 from the value of the predictor at the centroid of the regression, $\overline{x}$. The further from the centroid, the more any error in estimating the slope of the line will affect the prediction:

$$s^2_{Y_0} = s^2_{Y.x}\left[1 + \frac{1}{n} + \frac{(\mathbf{x}_0 - \overline{\mathbf{x}})^2}{\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})^2}\right] \tag{5}$$

where $\mathbf{x}$ refers to both coördinates.

The variance of the regression $s^2_{Y.x}$ is computed from the squared deviations of actual (y_i) and estimated ($\hat{y}_i$ values:

$$s^2_{Y.x} = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

To display a map of the interpolated surface, it's easiest to format the grid as a spatial object, so that the plotting method `spplot`'spatial plot" can be used.

Task 23 : Convert the grid to a spatial object: a `SpatialGridDataFrame`. •

The `gridded` function specifies that the spatial object is points on a regular grid. Since this is a complete grid, we can improve computational efficiency by using the `fullgrid` function to specify that the grid is complete ("full").

```
> coordinates(grid) <- c("e", "n")
> sp.grid <- SpatialPointsDataFrame(coords = coordinates(grid),
+     data = as.data.frame(pred.ts2))
> gridded(sp.grid) <- TRUE
> fullgrid(sp.grid) <- TRUE
> summary(sp.grid)
```

```
Object of class SpatialGridDataFrame
Coordinates:
    min  max
e -33.5 42.5
n -47.5 52.5
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
e               -33        1        76
n               -47        1       100
Data attributes:
      fit              lwr              upr
 Min.   :461.07   Min.   :448.96   Min.   :473.17
 1st Qu.:516.41   1st Qu.:505.15   1st Qu.:527.72
 Median :547.31   Median :536.06   Median :558.56
 Mean   :546.68   Mean   :535.39   Mean   :557.97
 3rd Qu.:577.01   3rd Qu.:565.73   3rd Qu.:588.22
 Max.   :614.63   Max.   :603.23   Max.   :626.03
```

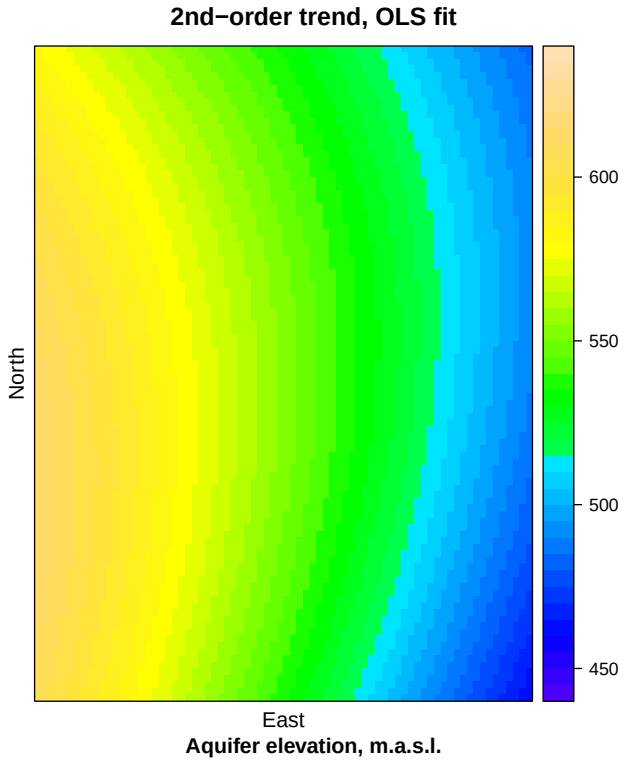
Task 24 : Display the best-fit interpolation, with the data points superimposed. •

The `spplot` "spatial plot" method plots spatial objects, i.e., those in one of the `sp` classes.

The `fit` field of the prediction object contains the trend surface fits.

We save this plot for comparison later with the Generalized Least Squares (GLS) trend surface (§7).

```
> ts.plot.breaks <- seq(440, 640, by=5)
> p.ols <- spplot(sp.grid, zcol="fit",
+              main="2nd-order trend, OLS fit",
+              sub="Aquifer elevation, m.a.s.l.",
+              xlab="East", ylab="North",
+              at=ts.plot.breaks,
+      col.regions = topo.colors(length(ts.plot.breaks)),
+      panel=function(x, ...) {
+             panel.levelplot(x, ...);
+             panel.points(coordinates(aq), pch=1,
+                 col=ifelse(res.ts2 < 0, "red", "black"),
+                 cex=2*abs(res.ts2)/max(abs(res.ts2)))
+ })

> print(p.ols)
```



In this plot the *residual* from the model at each observation point is shown (1) in colour: red = negative (actual < predicted), black = positive (actual > predicted). If a prediction is exactly on the trend surface it will not appear. This gives a nice visualization of the fit of the trend surface to the sample points.

> **Note:** The `spplot` method in the `sp` package makes use of the `levelplot` method of the `lattice` graphics package. Unlike base graphics, in `lattice` all plotting must be done at once; you can't start a plot and add more later. Graphical elements are added with a **panel function**, introduced with the `panel` argument. This function may contain many methods to draw graphic elements. In this case there is `panel.levelplot` to draw the levelplot (trend surface) and `panel.points` to place a set of points on top of it.

Q18 : *How well does the trend surface fit the points? Are there obvious problems?* *Jump to A18* •

Task 25 : Summarize the uncertainty from the trend surface, as absolute differences between the upper and lower prediction limits, and then this as a percentage of the best fit value. •

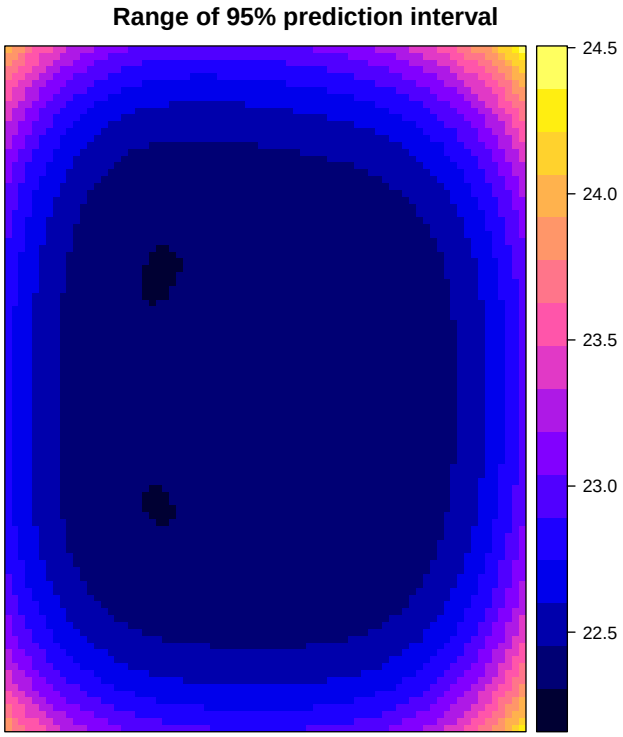
```
> summary(sp.grid$lwr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 448.96  505.15  536.06  535.39  565.73  603.23

> summary(sp.grid$diff <- sp.grid$upr - sp.grid$lwr)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 22.303  22.336  22.455  22.580  22.731  24.362

> summary(100 * sp.grid$diff/sp.grid$fit)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 3.7062  3.9004  4.1011  4.1500  4.3576  5.2525
```

The `lwr` "lower" and `upr` "upper" fields of the prediction object contain the lower and upper limits of the 95% *prediction interval* for each point on the grid. Their difference is the range of uncertainty; this divided by the fit is an approximation to 2 standard deviations.

Task 26 : Display the prediction interval of the trend surface as a map, showing also the location of the observation points. •

```
> spplot(sp.grid, zcol="diff",
+        main="Range of 95% prediction interval",
+        panel=function(x, ...) {
+            panel.levelplot(x, ...);
+            panel.points(coordinates(aq),
+            pch=20, col="white")}
+ )
```



Q19 : *What are the units of prediction interval? How large are they? How does this compare to the variable we are trying to predict?* *Jump to A19* •

Q20 : *Describe the spatial pattern of the prediction interval.* *Jump to A20* •

6 Spatial correlation of the residuals

We saw that the residuals from the OLS fit are not spatially independent – there are local clusters of similar values. The spatial structure of the **residuals** can be modelled with a **variogram**; this structure can be used to adjust the trend surface with Generalized Least Squares (GLS). In this section we examine the empirical variogram of the residuals and model it; this model can be used to improve the linear model and also can be used to make a better prediction, based on nearby values, using Simple Kriging (SK) on the residuals.

6.1 Extracting the residuals

Task 27 : Add the second-order trend-surface **predictions** and **residuals** as fields to the `aq` data frame. •

The `fitted` method extracts fitted values from a linear model object; the `residuals` method extracts the residuals.

```
> aq$fit.ts2 <- fitted(model.ts2)
> aq$res.ts2 <- residuals(model.ts2)
> str(aq)

'data.frame':        161 obs. of  8 variables:
 $ UTM.E  : num  569464 573151 559974 553514 550350 ...
 $ UTM.N  : num  4172115 4167193 4169585 4174584 4171337 ...
 $ z      : num  1628 1589 1676 1690 1691 ...
 $ zm     : num  496 484 511 515 516 ...
 $ e      : num  36.1 39.8 26.6 20.1 17 ...
 $ n      : num  -25.1 -30 -27.7 -22.7 -25.9 ...
 $ fit.ts2: num  490 480 507 521 526 ...
 $ res.ts2: num  6.1 4.59 4.2 -6.24 -10 ...
```

6.2 Making a spatial object

For this section we need to make the dataset into an explicitly *spatial* data structure. A **spatial object**, for the `sp` package, is one that has explicit coördinates. The `aq` dataframe does have coördinates, but "hidden" as attributes. These in fact have a special status. To continue the analysis, we identify these explicitly as being spatial.

Task 28 : Convert the dataset into a spatial object. •

The `coordinates` method specifies coördinates, thus converting a dataframe or matrix into an explicitly spatial object.

```
> coordinates(aq) <- c("e", "n")

> str(aq)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        161 obs. of  6 variables:
  .. ..$ UTM.E  : num [1:161] 569464 573151 559974 553514 550350 ...
  .. ..$ UTM.N  : num [1:161] 4172115 4167193 4169585 4174584 4171337 ...
  .. ..$ z      : num [1:161] 1628 1589 1676 1690 1691 ...
  .. ..$ zm     : num [1:161] 496 484 511 515 516 ...
  .. ..$ fit.ts2: num [1:161] 490 480 507 521 526 ...
  .. ..$ res.ts2: num [1:161] 6.1 4.59 4.2 -6.24 -10 ...
  ..@ coords.nrs : int [1:2] 5 6
  ..@ coords     : num [1:161, 1:2] 36.1 39.8 26.6 20.1 17 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "e" "n"
  ..@ bbox       : num [1:2, 1:2] -33 -47 41.1 51.1
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "e" "n"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr NA
```

This structure display is quite different from the previous one. The object now is of class `SpatialPointsDataFrame` and has five **slots**, marked with the `@` symbol.

The information in the original dataframe is now clearly split into two kinds:

Geographic space : Coordinates; location of the observation in some coördinate reference system;

Feature-space : Also called *attribute space*: properties of the observation. Here there is only one, the aquifer elevation.

Q21 : *Looking at the names of the slots, which likely refer to geographic space? Which slot contains the feature-space information?* *Jump to A21* •

We've done some work to get this data set into proper form for spatial analysis; so we save it in this format.

Task 29 : Save the spatial object as an R Data file. •

```
> save(aq, file = "aquifer.rda")
```

This can be read into a later R session with the `load` method.

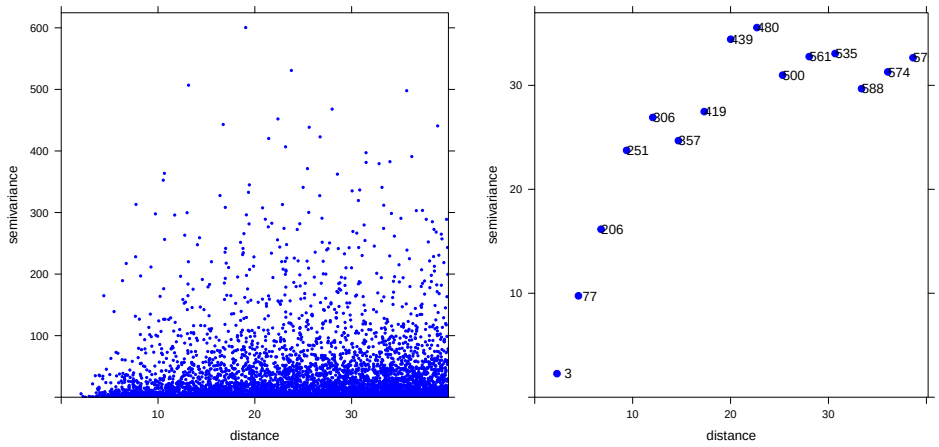
6.3 Omnidirectional (isotropic) variogram analysis

Task 30 : Compute and plot the **omnidirectional empirical variogram** of

the residuals from the second-order surface, with a cutoff of 40 km. •

The `variogram` function of the `gstat` package computes the empirical variogram We show both the *variogram cloud* and the *summarized variogram*, which averages the points in the variogram cloud over some separation ranges; these are called **variogram bins**.

```
> vr.c <- variogram(res.ts2 ~ 1, loc = aq, cutoff = 40,
+     cloud = T)
> vr <- variogram(res.ts2 ~ 1, loc = aq, cutoff = 40)
> p1 <- plot(vr.c, col = "blue", pch = 20, cex = 0.5)
> p2 <- plot(vr, plot.numbers = T, col = "blue", pch = 20,
+     cex = 1.5)
> print(p1, split = c(1, 1, 2, 1), more = T)
> print(p2, split = c(2, 1, 2, 1), more = F)
```



Note: The code to print two variograms side-by-side uses the **split** and **more** optional arguments to the **print** method for Lattice graphics plots.

Q22 : *What are the estimated sill, range, and nugget of this variogram?* *Jump to A22* •

The **shape** of the variogram suggests that an **exponential** model would fit it[7].

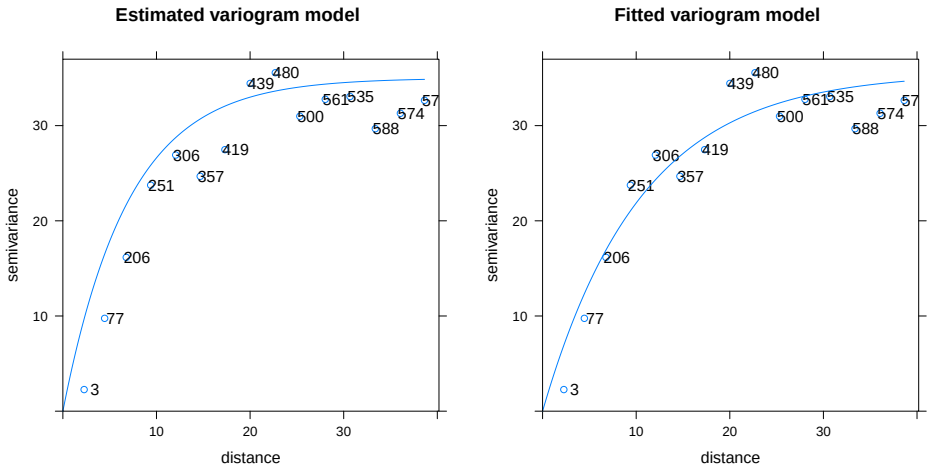
Task 31 : Model this variogram with an exponential model by eye, and then fit it with `gstat`'s default automatic fit. Plot both models side-by-side. •

Note: An exponential model's **effective range** (where it reaches 95% of its asymptotic sill) is three times the **range parameter** of the variogram model. Thus in this case if we've estimated 21 km range, we specify 7 km as our initial guess for the range parameter.

[7] There are many model shapes; in this introduction there is no space to discuss them

The `fit.variogram` function of the `gstat` package uses weighted least squares to adjust the variogram model to the empirical variogram.

```
> vr.m <- vgm(35, "Exp", 7, 0)
> (vr.m.f <- fit.variogram(vr, vr.m))

  model     psill    range
1   Nug  0.000000  0.00000
2   Exp 35.551201 10.47154

> p1 <- plot(vr, plot.numbers=T,
+            model=vr.m, main="Estimated variogram model")
> p2 <- plot(vr, plot.numbers=T,
+            model=vr.m.f, main="Fitted variogram model")
> print(p1, split=c(1,1,2,1), more=T)
> print(p2, split=c(2,1,2,1), more=F)
```



Q23 : *What are the parameters of the fitted variogram?* *Jump to A23* •

7 Trend surface analysis by Generalized Least Squares

As explained in §4, the OLS solution is only valid for independent residuals. The previous § shows that in this case the residuals are *not* spatially independent, and we were able to model that dependence with a variogram model. Thus, using OLS may result in an incorrect trend surface equation, although the OLS estimate is unbiased. A large number of close-by points with similar values will "pull" a trend surface towards them. Furthermore, the OLS R^2 (goodness-of-fit) may be over-optimistic. This is discussed by Fox [3, §14.1].

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]:

$$\hat{\beta}_{\text{gls}} = (X^T C^{-1} X)^{-1} X^T C^{-1} y \tag{7}$$

where X is the design matrix, C the **covariance matrix** of the (spatially-correlated) **residuals**, and y the vector of **observations**. If there is no spatial dependence among the errors, C reduces to $I\sigma^2$ and the estimate to OLS as in Equation 3.

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 the covariance between all the point-pairs, since we only have one realization of the random field. So we model the covariance as a function of the **separation** (usually the distance) between point pairs, similar to what we did in §6.3, 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.

One method to compute the GLS model is iterative:

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;
5. repeat steps (2)–(4) until the covariance structure does not change between iterations.

In many cases only one iteration is necessary. However, theoretically this is not optimal, because the estimates of the covariance parameters are biased.

A more elegant solution is to fit the covariance structure at the same time the trend surface coefficients are computed.

7.1 Computing the GLS trend surface

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. A better solution is to compute the trend and the covariance at the same time, using Residual maximum likelihood (REML). See Lark and Cullis [5] for the mathematical development. This is implemented in the `gls` method of the `nlme` package.

Task 32 : Compute the coefficients of a full second-order trend, using GLS. •

```
> require(nlme)
> model.ts2.gls <- gls(
+    model = zm ~ n + e + I(n^2) + I(e^2) + I(e * n),
+    data = aq,
+    method="ML",
+    correlation=corExp(form=~e + n,
+       nugget=FALSE,
+       value=c(vr.m.f[2,"range"]))
+  )
> class(model.ts2.gls)

[1] "gls"

> summary(model.ts2.gls)

Generalized least squares fit by maximum likelihood
  Model: zm ~ n + e + I(n^2) + I(e^2) + I(e * n)
  Data: aq
        AIC       BIC     logLik
  939.38904 964.04028 -461.69452

Correlation Structure: Exponential spatial correlation
 Formula: ~e + n
 Parameter estimate(s):
    range
14.421573

Coefficients:
                Value Std.Error    t-value p-value
(Intercept) 559.30664 3.3622674 166.348052  0.0000
n            -0.05110 0.0546869  -0.934399  0.3516
e            -1.54989 0.0675256 -22.952609  0.0000
I(n^2)       -0.00547 0.0016897  -3.236521  0.0015
I(e^2)       -0.00125 0.0024507  -0.511079  0.6100
I(e * n)      0.00453 0.0019477   2.325551  0.0213

 Correlation:
         (Intr) n      e      I(n^2) I(e^2)
n         0.017
e         0.057 -0.038
I(n^2)   -0.620 -0.083  0.024
I(e^2)   -0.571  0.029 -0.233  0.099
I(e * n)  0.031 -0.048  0.012 -0.026 -0.066

Standardized residuals:
        Min          Q1         Med          Q3         Max
-3.40969259 -0.54853371  0.12769732  0.65139981  2.07524940

Residual standard error: 6.385512
Degrees of freedom: 161 total; 155 residual
```

Notice that the `gls` method also estimates the range of spatial correlation.

Q24 : *What is the range of spatial correlation of the exponential model,*

as estimated by `gls`*? Does this agree with the estimate from fitting the variogram of the residuals? What could account for the difference?* *Jump to A24* •

This gives different coefficients than the OLS fit.

Task 33 : Compare the coefficients from the GLS and OLS fits, as absolute differences and as percentages of the OLS fit. •

The generic `coef` method extracts coefficients from model objects.

```
> coef(model.ts2.gls) - coef(model.ts2)

   (Intercept)             n             e         I(n^2)
-1.75094930811 -0.03454451954  0.07085948553  0.00203106936
        I(e^2)       I(e * n)
 0.00039507895 -0.00217053276

> round(100 * (coef(model.ts2.gls) - coef(model.ts2))/coef(model.ts2),
+     1)

(Intercept)           n           e      I(n^2)      I(e^2)
       -0.3       208.7        -4.4       -27.1       -24.0
   I(e * n)
      -32.4
```

Q25 : *Why are the GLS coefficients different than the OLS coefficients?* *Jump to A25* •

Task 34 : Display the 90% confidence intervals for the GLS model parameters. •

The generic `intervals` method has a specific method for a fitted GLS model; internally this is the `intervals.gls` function of the `nlme` package.

```
> intervals(model.ts2.gls, level = 0.9)

Approximate 90% confidence intervals

 Coefficients:
                     lower           est.          upper
(Intercept) 553.7429446750 559.3066357528 564.8703268306
n            -0.1415921569  -0.0510993632   0.0393934305
e            -1.6616273076  -1.5498896721  -1.4381520365
I(n^2)       -0.0082645545  -0.0054686072  -0.0026726598
I(e^2)       -0.0053077229  -0.0012524867   0.0028027495
I(e * n)      0.0013065100   0.0045294039   0.0077522979
attr(,"label")
[1] "Coefficients:"

 Correlation structure:
         lower      est.     upper
```

```
range 8.1391394 14.421573 25.553287
attr(,"label")
[1] "Correlation structure:"

 Residual standard error:
    lower      est.     upper
5.0059190 6.3855120 8.1453104
```

Q26 : *Are the OLS estimates of the trend surface parameters, and the spatial correlation range parameter of the empirical variogram from these residuals, within the 90% confidence intervals from the GLS model?* *Jump to A26* •

7.2 Predicting from the GLS trend surface

Task 35 : Predict over the grid with the GLS trend. •

The `predict` generic method has a specific method for a fitted GLS model; internally this is the `predict.gls` function of the `nlme` package.

```
> pred.ts2.gls <- predict(model.ts2.gls, newdata=grid)
> summary(pred.ts2.gls)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 473.38  517.83  547.48  547.04  576.25  610.93
```

Task 36 : Display the best-fit interpolation, with the data points superimposed. •

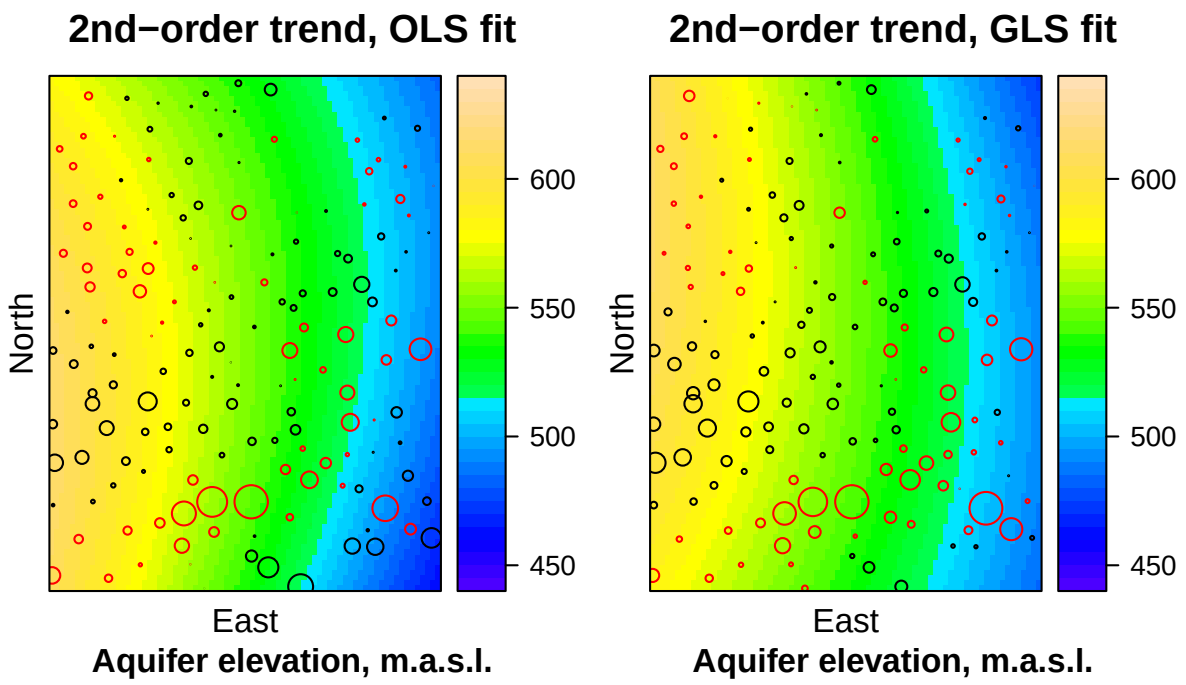
First we need to compute and store the residuals, to be displayed on the trend surface:

```
> res.ts2.gls <- residuals(model.ts2.gls)
```

The `spplot` "spatial plot" method plots spatial objects, i.e., those in one of the `sp` classes.

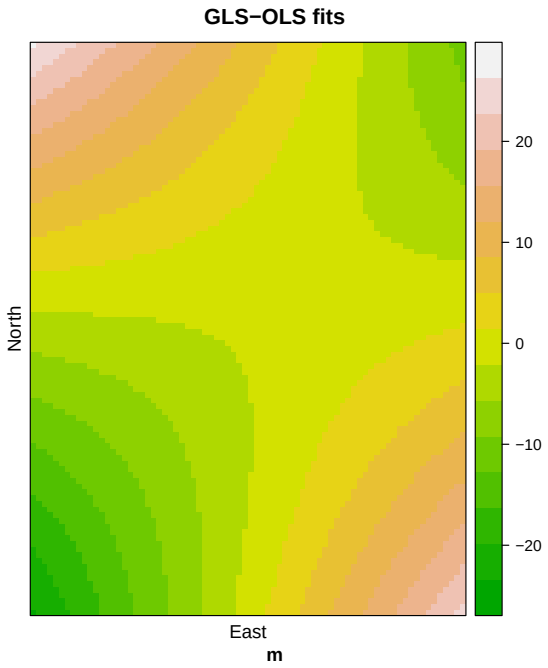
The `fit` field of the prediction object contains the trend surface fits.

```
> sp.grid$gls.fit <- pred.ts2.gls
> p.gls <- spplot(sp.grid, zcol="gls.fit",
+               main="2nd-order trend, GLS fit",
+               sub="Aquifer elevation, m.a.s.l.",
+               xlab="East", ylab="North",
+               at=ts.plot.breaks,
+      col.regions = topo.colors(length(ts.plot.breaks)),
+      panel=function(x, ...) {
+             panel.levelplot(x, ...);
+             panel.points(coordinates(aq), pch=1,
+                 col=ifelse(res.ts2.gls < 0, "red", "black"),
+                 cex=2*abs(res.ts2.gls)/max(abs(res.ts2.gls)))
+ })
> print(p.ols, split=c(1,1,2,1), more=T)
> print(p.gls, split=c(2,1,2,1), more=F)
```



Task 37 : Compute the difference between the OLS and GLS trend surfaces, and map them. •

```
> sp.grid$diff <- sp.grid$gls.fit - sp.grid$fit
> spplot(sp.grid,   zcol="diff",  main="GLS-OLS fits",
+              sub="m", xlab="East", ylab="North",
+              col.regions = terrain.colors(64))
```



Q27 : *Where are the largest differences between the OLS and GLS trend surfaces? Explain why.* *Jump to A27* •

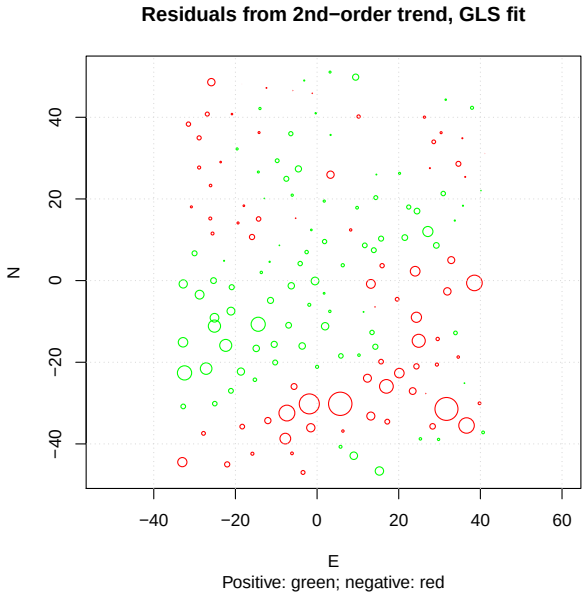
8 Local interpolation of the residuals

The trend surface fits an overall trend, but of course does not fit every observation exactly. This lack of fit can be pure noise, but it can also have a spatially-correlated component which can be modelled and used to improve the predictions.

Task 38 : Display the residuals from the GLS trend surface as a postplot. •

```
> summary(res.ts2.gls)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-21.772633  -3.502669   0.815413  -0.057213   4.159521  13.251530

> plot(aq$n ~ aq$e, cex=3*abs(res.ts2.gls)/max(abs(res.ts2.gls)),
+      col=ifelse(res.ts2.gls > 0, "green", "red"),
+      xlab="E", ylab="N",
```

```
+      main="Residuals from 2nd-order trend, GLS fit",
+      sub="Positive: green; negative: red", asp=1)
> grid()
```

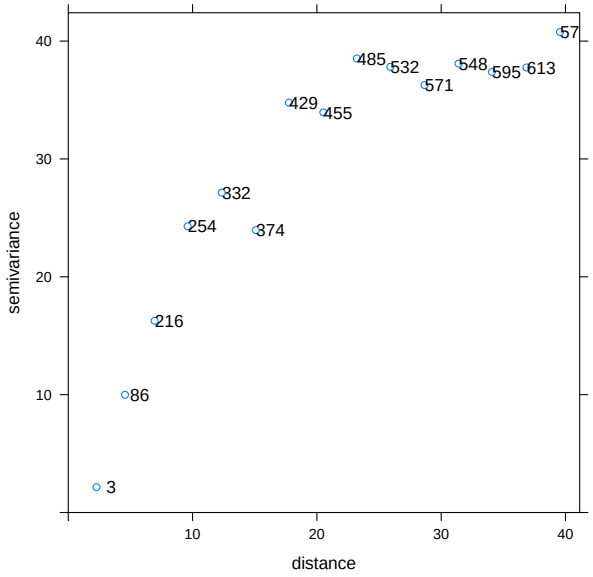


We can see from this post-plot of the residuals that there is local spatial correlation. The GLS fit optimized the estimates of the trend surface coefficients, and correctly estimated the spatial correlation of the residuals, but did not correct for this in mapping.

Task 39 : Compute the empirical variogram model residuals from the GLS trend surface model, and fit it with an exponential model. •

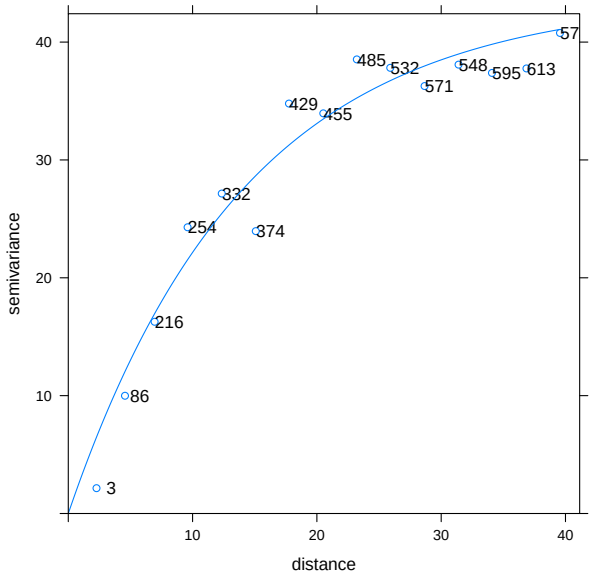
First extract the residuals into the point observations object, compute the empirical variogram, and display it to estimate the variogram model parameters.

```
> aq$res.ts2.gls <- residuals(model.ts2.gls)
> vr.gls <- variogram(res.ts2.gls ~ 1, loc = aq)
> plot(vr.gls, plot.numbers = T)
> (vr.gls.m.f <- fit.variogram(vr.gls, vgm(35, "Exp", 7,
+     0)))

  model     psill     range
1   Nug  0.000000  0.000000
2   Exp 43.758464 14.169087
```



Second, fit it and display the fitted model on the empirical variogram.

```
> (vr.gls.m.f <- fit.variogram(vr.gls, vgm(35, "Exp", 7,
+     0)))

  model     psill     range
1   Nug  0.000000  0.000000
2   Exp 43.758464 14.169087

> plot(vr.gls, model = vr.gls.m.f, plot.numbers = T)
```



Q28 : *How does this variogram model compare to the variogram computed from the OLS 2nd-order surface (§6.3)? Does the range parameter of this*

model agree with the estimate from the GLS fit? *Jump to A28* •

```
> vr.m.f

  model     psill    range
1   Nug  0.000000  0.00000
2   Exp 35.551201 10.47154

> vr.gls.m.f

  model     psill     range
1   Nug  0.000000  0.000000
2   Exp 43.758464 14.169087

> intervals(model.ts2.gls)$corStruct[2]

[1] 14.421573
```

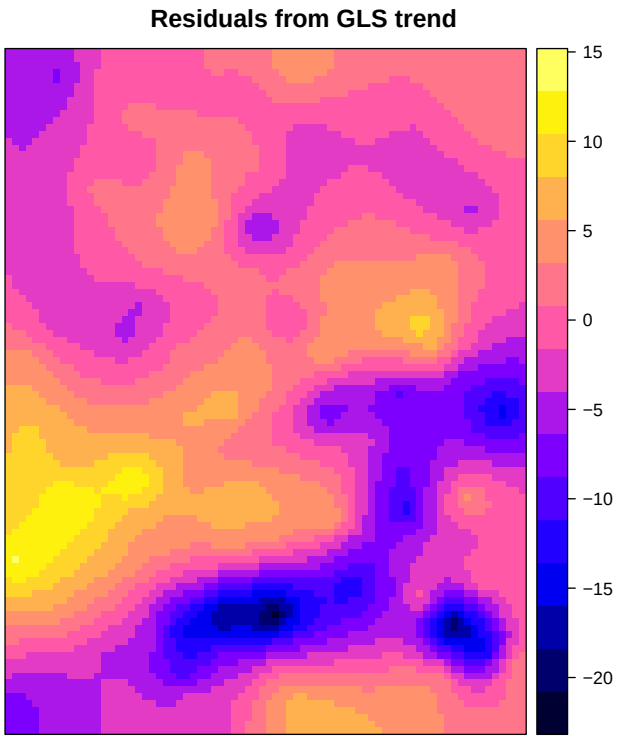
Task 40 : Interpolate the residuals onto the prediction grid by Ordinary Kriging (OK). •

```
> kr <- krige(res.ts2.gls ~ 1, loc = aq, newdata = sp.grid,
+     model = vr.gls.m.f)

[using ordinary kriging]

> summary(kr)

Object of class SpatialGridDataFrame
Coordinates:
    min  max
e -33.5 42.5
n -47.5 52.5
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
e               -33        1        76
n               -47        1       100
Data attributes:
   var1.pred             var1.var
 Min.   :-20.824450   Min.   : 0.13688
 1st Qu.: -3.141128   1st Qu.: 7.96014
 Median : -0.037762   Median :10.20159
 Mean   : -0.372732   Mean   :10.83563
 3rd Qu.:  2.783062   3rd Qu.:12.43497
 Max.   : 12.832161   Max.   :33.72589
```

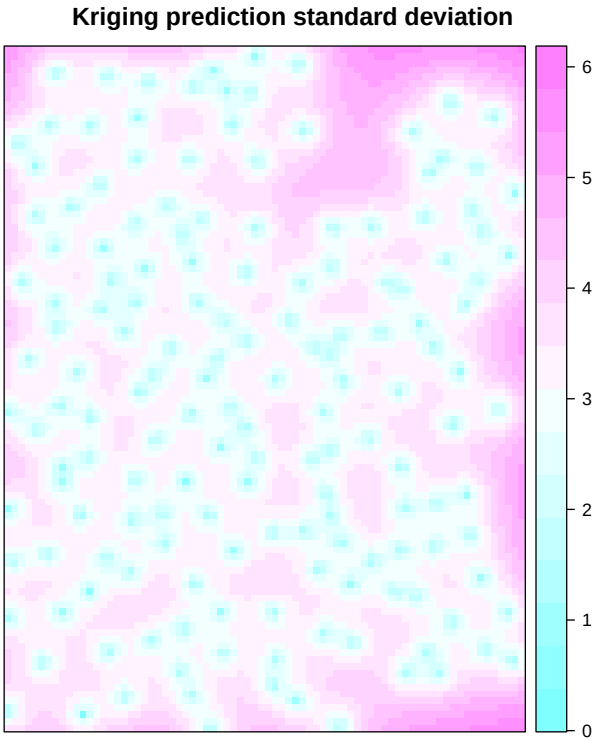
Note: Notice that the mean kriging prediction is not zero.

Task 41 : Display the kriging predictions and their prediction standard deviations. •

```
> p1 <- spplot(kr, zcol = "var1.pred", col.regions = bpy.colors(64),
+     main = "Residuals from GLS trend")
> kr$var1.sd <- sqrt(kr$var1.var)
> print(p1)
```




```
> p2 <- spplot(kr, zcol = "var1.sd", col.regions = cm.colors(64),
+     main = "Kriging prediction standard deviation")
> print(p2)
```



Q29 : *Which areas were most changed by interpolating the residuals? Why? Which areas have the most and least uncertainty? Why?* *Jump to A29* •

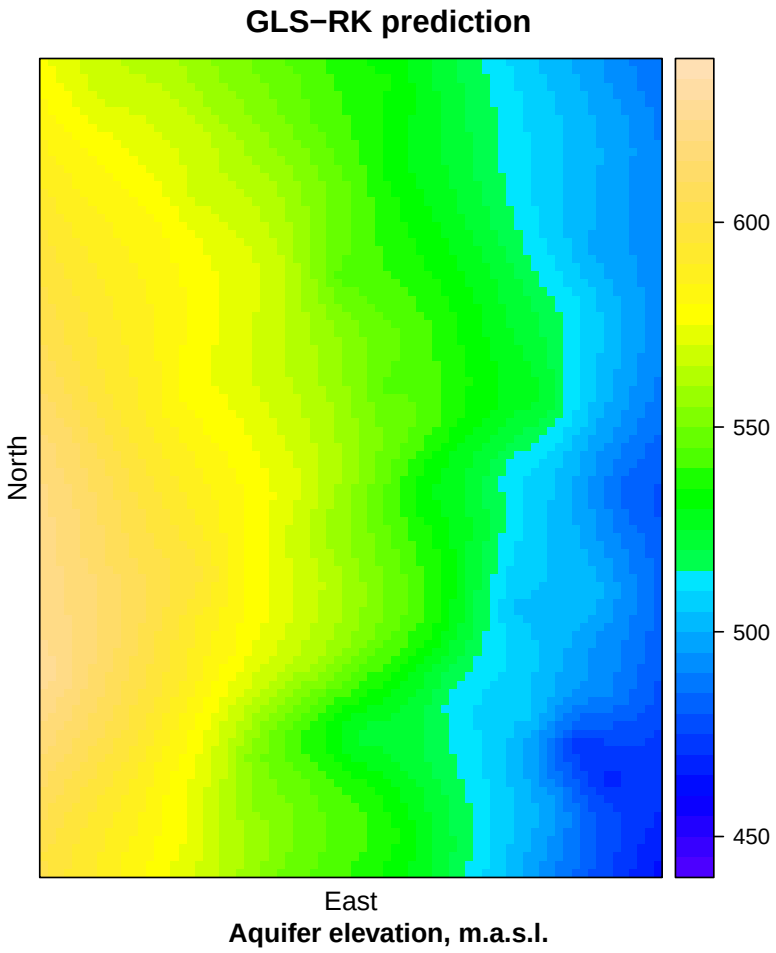
Task 42 : Add the OK predictions of the residuals to the prediction grid object, and then add them together with the trend surface prediction to obtain a final prediction. •

The kriging prediction object was built from the spatial grid, so it has the same dimensions.

```
> sp.grid$ok <- kr$var1.pred
> sp.grid$rk.gls <- sp.grid$fit + sp.grid$ok
```

Task 43 : Plot the final prediction. •

```
> p.rk <- spplot(sp.grid, zcol="rk.gls",
+              main="GLS-RK prediction",
+              sub="Aquifer elevation, m.a.s.l.",
+              xlab="East", ylab="North",
+              at=ts.plot.breaks,
+      col.regions = topo.colors(length(ts.plot.breaks)))
> print(p.rk)
```



9 Cleaning up

Task 44 : Remove the temporary objects from the workspace. Leave the fitted variogram model and the spatial points data frame with the trend surface results. •

```
> rm(p1, p2)
```

Task 45 : Save the workspace. •

```
> save.image(file = "tsresults.RData")
```

This saves the workspace (objects and their names), in R's binary format.

Task 46 : Quit R. •

9.1 Answers

A1 : *The map of aquifer elevations, along with a map of the elevation of the land surface, can be used by well-drillers, to estimate the cost of drilling a well to reach the aquifer at any location.* *Return to Q1* •

A2 : *There are 161 observations (wells); for each we know the coördinates (`E` and `N`) and the elevation of aquifer (`z`); we also have the transformed elevation in meters and the reduced coördinates.* *Return to Q2* •

A3 : *UTM East from 500361.3 m . . . 574429.6 m (range 74.068 km); UTM North from 4150248.2 m . . . 4248312.5 m (range 98.064 km); total area 7263 km².* *Return to Q3* •

A4 : *Elevations are from 476 to 623 m.a.s.l., a range of 148 m.* *Return to Q4* •

A5 : *Nearby points tend to be similar; there appears to be trend from E to W, but there are portions of the map that do not follow this strictly.* *Return to Q5* •

A6 : *(1) The text postplot has the advantage of showing the actual values, but it is not very graphical and difficult to read; (2) the size postplot clearly shows the relative data values; (3) the size and colour postplot gives two ways to visualize; it seems especially good for seeing the E–W increasing first-order trend.* *Return to Q6* •

A7 : *The aquifer has a flat surface, tilted towards some direction, by some regional uplift. In this case, the uplift of the Rocky Mountains about 650 km to the west has tilted the aquifer.* *Return to Q7* •

A8 : *The trend surface equation is:* z = 555 + -1.617135 e + -0.033361 n. *The* intercept *term gives the estimated aquifer elevation at the centroid of the area. Then the two* coefficients *give the change in elevation per unit change of the target variable. That is, for each km E the elevation decreases by -1.62 m, for each km N it decreases by -0.03 m. The relation is highly-significant; it explains 94.1% of the variability in the observations; however the N coördinate is not needed – it is not statistically different from zero.* *Return to Q8* •

A9 : *Residuals range from -25.4 to 16.7 m; compare this to the median elevation 552.8 m; the maximum calibration error is 4.6%.* *Return to Q9* •

A10 :

1. *No relation between fitted values and residuals; but . . .*

2. *The residuals are not normally-distributed, especially in the high tail. That is, the largest positive residuals (under-predictions) are not as extreme as would be expected. The largest negative residuals (over-predictions) are a bit too extreme.*

Conclusion: this OLS fit does not satisfy the assumptions of independent residuals.
Return to Q10 •

A11 : *There is a spatial pattern. Large residuals tend to be near each other, and vice-versa. Positive residuals (above the trend surface) are found almost exclusively in the middle third of the map. Dependence seems to be stronger along a SW-NE axis (range about 50 to 70 km) than the NW-SE axis (range about 10 to 20 km). This implies a higher-order trend surface or a periodic surface superimposed on the linear trend.*
Return to Q11 •

A12 : *The tilted structure has local warping as either a dome or a basin.* *Return to Q12* •

A13 : *The model explains 97.5% of the variance in the observations, compared to 94.1% for the first-order significance.*
Return to Q13 •

A14 : *The probability that the higher-order surface is this much better just by chance is almost zero, so the second-order surface is statistically superior to the first-order surface.*
Return to Q14 •

A15 : *Residuals range from -19.8 to 14.8 m; compare this to the median elevation 552.8 m; the maximum calibration error is 3.6%. This range is narrower than for the first-order surface: -25.4 to 16.7 m.*
Return to Q15 •

A16 :

1. *No relation between fitted values and residuals;*
2. *The residuals are not normally-distributed. The largest negative residuals (over-predictions) are a bit too extreme. However, the problem with the largest positive residuals from the first-order surface has been solved.*

Conclusion: this OLS is much closer to being valid than for the first-order surface.
Return to Q16 •

A17 : *These residuals form local clusters of positive, negative, and near-zero; there does not appear to be any overall spatial pattern. So, a higher-order trend surface is not indicated. Instead, some local interpolation of the residuals would seem to improve the model.*
Return to Q17 •

A18 : *The fit is generally good but some clusters of points stand out from the*

background; their values are not that well matched. *Return to Q18* •

A19 : *The prediction errors are from -23.5 to 26.3 m; this is about -0.1% of the predicted value. This much uncertainty in the prediction corresponds to uncertainty in the expense of drilling a well at the location.* *Return to Q19* •

A20 : *They are least at the centre of gravity of the regression in both E and N; they increase away from this in both directions; the largest uncertainties are in the corners of the grid.* *Return to Q20* •

A21 : *There are several* ***slots*** *in the object that refer to geographic space:*

1. `bbox` *for the bounding box (extreme values of coördinates)*
2. `coords` *storing the coördinatesof each observation*
3. `proj4string` *for the map projection, not used here*

The attribute data are in slot `data`*, which is a data frame, like the original (non-spatial) dataset. In this case there is only one attribute: the elevation of the aquifer at the location.* *Return to Q21* •

A22 : *The sill of about* 35 m^2 *is reached near a range of* 21 *km; there is no evidence of nugget variance.* *Return to Q22* •

A23 : *Exponential model: total sill 36* m^2*, distance parameter 10 m, so the effective range is 31 m; nugget variance 0* m^2*. A zero or very small nugget is to be expected with spatially-continuous variables like groundwater level; the only reason for a non-zero nugget would be measurement error.* *Return to Q23* •

A24 : *The range parameter of spatial correlation of the exponential model, as estimated by* `gls`*, is 14.4 km. This does not agree with the estimate from fitting the variogram of the residuals, . These two fits use completely different methods. The* `fit.variogram` *method uses weighted least squares on the binned empirical variogram, the weights proportional to* n/h^2*. Also, the empirical variogram is of residuals computed by OLS. The* `gls` *method uses restricted maximum likelihood (REML) on the entire dataset, which does not depend on a variogram nor on the OLS fit.* *Return to Q24* •

A25 : *High and low residuals from the OLS fit are clustered. This shows that the OLS trend surface is being "pulled" towards these highest and lowest values, which are on the edges of map, thus leading to a more strongly "tilted" surface. The GLS fit corrects for this by effectively declustering the correlated residuals.* *Return to Q25* •

A26 : *Yes.* *Return to Q26* •

A27 : *The largest positive differences (GLS higher than OLS) are in the NW and SE corners; the largest negative differences in the NE and SW corners. This is because all the coefficients, which are E, N or combinations, are lower for GLS than for OLS. So in the SW–NE plane the values are lower; in the opposite NW–SE plane higher.* Return to Q27 •

A28 : *The variogram from the GLS residuals has a somewhat higher sill and quite a bit longer range parameter, about 14 km vs. about 10.5 km The fitted range agrees closely with that fitted as part of the GLS procedure.* Return to Q28 •

A29 : *The centre-SE has a large negative adjustment, the centre-SW a large positive adjustment. These are the areas with clusters of model residuals of the corresponding sign.*

The most certain predictions are near the observation points, especially where they are clustered. The least certain are in the NE and SE corners, where there are few observations. Return to Q29 •

A Derivation of the OLS solution to the linear model

To solve Equation 2 we need an **optimization criterion**, i.e., what makes a particular solution (values of β) better than any other. The obvious criterion is to minimize the total error (lack of fit) as some function of $\varepsilon = \mathbf{y} - \mathbf{X}\beta$; the goodness-of-fit is then measured by the size of this error. A common way to measure the total error is by the sum of vector norms; in the simplest case the Euclidean distance from the expected value, which we take to be 0 in order to have an unbiased estimate. If we decide that both positive and negative residuals are equally important, and that larger errors are more serious than smaller, the vector norm is expressed as the sum of squared errors, which in matrix algebra can be written as:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \tag{8}$$

which expands to

$$\begin{aligned} S &= \mathbf{y}^T\mathbf{y} - \beta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta \\ S &= \mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta \end{aligned} \tag{9}$$

> **Note:** $\mathbf{y}^T\mathbf{X}\beta$ is a 1×1 matrix, i.e., a scalar[8], so it is equivalent to its transpose: $\mathbf{y}^T\mathbf{X}\beta = [\mathbf{y}^T\mathbf{X}\beta]^T = \beta^T\mathbf{X}^T\mathbf{y}$. So we can collected the two identical 1×1 matrices (scalars) into one term.

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

$$\begin{aligned} \frac{\partial}{\partial \beta^T} S &= -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta \\ \mathbf{0} &= -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\beta \\ (\mathbf{X}^T\mathbf{X})\beta &= \mathbf{X}^T\mathbf{y} \\ (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X})\beta &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ \hat{\beta}_{\text{OLS}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{10}$$

which is the OLS solution.

B Standardized residuals

Standardized residuals[9] adjust the residuals from a linear regression model to residuals which should be distributed as $\mathcal{N}(0, 1)$ with equal variance. These can then be compared to residuals drawn from that theoretical distribution, for example in a quantile-quantile ("QQ") plot of the standardized residuals.

The standardized residuals are computed as $r_i/(s \cdot \sqrt{1 - h_{ii}})$, where r_i are the unstandardized residuals, s is the sample standard deviation of the residuals, and the h_{ii} are the diagonal entries of the so-called "hat" matrix $V = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$.

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

[9] This is the term used by `plot.lm`; some authors call this the "studentized" residuals.

The sample standard deviation of the residuals s is computed as the square root of the estimated variance of the random error:

$$s = \sqrt{\frac{1}{(n-p)} \cdot \sum r_i^2}$$

where n is the number of observations and p the number of predictors. It is shown in the linear model summary as "Residual standard error"; it can be extracted as `summary(model_name)$sigma`. This is an overall measure of the variability of the residuals, and so can be used to standardize the residuals to $\mathcal{N}(0, 1)$.

The "hat" matrix V is another way to look at linear regression. This matrix multiplies the observed values to compute the fitted values. The hat value for an observation gives the overall leverage (i.e., importance when computing the fit) of that observation. So the term $\sqrt{1 - h_{ii}}$ in the denominator shows that with low influence (small h_{ii}) the ratio r_i/s (a simple standardization) is not affected much, but with a high influence (large h_{ii}) the denominator is smaller and so the standardized residual is increased. Thus the standardized residuals are higher for points with high influence on the regression coefficients.

References

[1] N Cressie. *Statistics for spatial data*. John Wiley & Sons, revised edition, 1993. 28

[2] J C Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 1, 4, 18

[3] J Fox. *Applied regression, linear models, and related methods*. Sage, Newbury Park, 1997. 27

[4] R Ihaka and R Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. 1

[5] R. M. Lark and B. R. Cullis. Model based analysis using reml for inference from systematically sampled data on soil. *European Journal of Soil Science*, 55(4):799–813, 2004. doi: 10.1111/j.1365-2389.2004.00637.x. 28

[6] R A Olea and J C Davis. Sampling analysis and mapping of water levels in the High Plains aquifer of Kansas. Technical Report KGS Open File Report 1999-11, Kansas Geological Survey, May 1999. URL http://www.kgs.ku.edu/Hydro/Levels/OFR99_11/. 1, 2

[7] R. A. Olea and John C. Davis. Optimization of the high plains aquifer water-level observation network. Technical Report KGS Open File Report 1999-15, Kansas Geological Survey, May 1999. URL http://www.kgs.ku.edu/Hydro/Levels/OFR99_15/. 1

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org. 1

[9] R Development Core Team. *R Data Import/Export*. The R Foundation for Statistical Computing, version 3.4.1 (2017-06-30) edition, 2017. URL http://cran.r-project.org/doc/manuals/R-data.pdf. 4