Spatial Analysis Techniques in R

Lesson 4: Geostatistics, IDW & field data

4.1 Introduction: the spatial Interpolation problem

Task 4.1: The Problem

Read Chapter 9, Sections 9.1-9.3 pages 239 -263 of O'Sullivan and Unwin (2010). Much the same material is in the First Edition, Chapter 8, pages 209 – 234. If you have the time you might attempt to little exercise *Spatial interpolation by Hand and Eye*, pages 251-253 (or 222-223 in the First Edition).

In my basic *statistics.com* course on spatial analysis I set students an assignment using the same exercise as that in Task 4.1 (above) and the results always show three things. First, a few students think that it is trivial, akin to those childhood 'join the dots' puzzles and is thus well beneath their dignity. Some get angry about having been asked to do it at all. Almost always such students produce rather poor maps. Second, at best 1 in 10 of all the resulting attempts produces a pattern of iso-therms (contours of equal temperature) that is even vaguely 'correct' in 'honoring' all the data and not having logically impossible isotherm patterns. Third, even among those that are technically correct, there is a considerable variation in the resulting predicted field of numbers.

In short the problem isn't simple. Basically, *interpolation* involves the use of a set of sample control points at which the location (*x* , *y*) and 'height ' (*z*) of a field of data is known to predict the value of *z* at all the unknown locations across the entire field. For the moment we will assume these *z*, *x* and *y* values are measured exactly but in many practical applications this will not be the case.

In Lesson (1) of this course, Section 1.6, we brought some (*x, y, z*) sample data for surface height in a small area (topo_data), for soils (meuse), and for atmospheric ozone (laozone) into R and visualized them using two very simple but 'honest' methods based on color coding of the point values and 'bubble' plots in which the bubble size is varied as a function of the height, or *z*, value. In the terms we use in the course text, we have *sampled* the field/surface and provided a *digital description* of it by way of a simple list of the z (height) values. Anything beyond this, any *interpolation*, whether by hand and eye or computer algorithm, creates a *model* of the field that may, or may not, correspond with reality on the ground. Sometimes we can check it, for example by re-survey or by using subsets of the available sample data, but in most applications we simply can't go back and collect more sample data and so have to make do with what we have. In a real sense in a spatial interpolation we have to try to 'know the unknown' and there are two general ways of tackling the problem. Mathematically, the problem can be regarded as a two-dimensional variation on curve fitting to model an unknown function. The classic approach would be to approximate the unknown f(*x, y*) by a parametric function whose general form is postulated in advance, either explicitly, as in a polynomial in *x* and *y* or as a weighted sum of the neighborhood values as in inverse distance weighting (IDW) approaches, or implicitly, as in the imposition of a minimum curvature condition. Parameters for these functions might be provided *a priori* (as in IDW) or estimated from the available sample data and, once the functions are computed they could be used to predict the unknown surface height at every location across the area of interest.

Statistically, there is another way of looking at the problem, which is to start with a model of what we see in nature rather than with a model of the interpolating function. This is the approach known as *kriging*, the term coined by Frenchman G. Matheron in 1963 after the name of D. O. Krige. In what is now over a half-century following Matheron's initial work, the basic ideas have been shown to be extremely useful in many fields (notably in the mining industry, geology and hydrology/meteorology) and have been developed to deal with a much larger variety of problems, with the entire field acquiring the somewhat misleading name of *geostatistics* (isn't all we have done so far in some sense 'geostatistics'? I think so, and anyhow what of problems that aren't 'geo' that can be addressed using the same methods?). The result of all this activity is that there is a huge literature with some really quite 'heavy' (both in weight and in the level of mathematics needed to follow them) texts. Most people suggest Isaaks and Srivastava (1989) as a good place to start. I very much like the clarity of the explanations in Webster and Oliver (2001) and often find that when I have a query the answer will be somewhere in the 695 pages of Chilẻs and Delfiner (1999).

In addition and at a much simpler mathematical level there is a free of charge text by Isobel Clarke (originally 1979) available for download at

http://www.kriging.com/pg1979_download.html (checked 11-12-15)

which contains all that you need to get started in geostatistics.

There are several R packages for handling and analyzing geostatistical data. The gstat package provides numerous uni- and multi-variate approaches and geoR and geoRglm provide functions for what is called *model-based geostatistics*. The fields package is similar, and the spatial package (part of the VR bundle in the base distribution) has some core functions. The RandomFields package has facilities for simulating and analyzing random fields. The package sgeostat is also available as are tripak and akima that can be used for deterministic, mathematical interpolation using triangulation and splines respectively.

In this lesson we can only scratch the surface of that is available, hoping that it provides an introduction into the entire field. Since most people using these approaches are interested in the interpolation of field data this is what we will concentrate on but throughout you should be aware that there is much, much more that could be added.

4.2 Mathematical Approaches

Most methods of interpolation, including spatial prediction using geostatistical approaches, adopt a two stage approach in which values of the field are estimated over a very fine grid (i.e. a *raster*) and then this grid is rendered and displayed in some way or other. Methods based on *contour chasing* through the Delauney triangulation of the point pattern are an exception but, although these were the original automated approach to the problem, these seem little used nowadays. In R, the designers of relevant packages were utterly logical in their approach so that to produce a contour map from a scatter of control points we have explicitly to undertake a series of steps:

a) Incorporate the sample data into R as a spatial data table;
b) Specify the grid onto which the interpolation is to be undertaken as a `grid` object and then create from this a pixel data frame;
c) Use whatever approach is required to perform the interpolation onto the grid and its pixel equivalent;
d) Finally, render the resulting object in some way for display, for example as a contour map, which involves creating a spatial line object or an image.

We will first explore this process using possibly the most frequently used two stage approach known as *inverse distance weighting* (IDW). Historically, this was the method used in SYMAP, which in the 1960s was the first widely used interpolation routine for computer contouring. What I would like you to do here is Task 4.2 and it forms the first part of your assignment for this lesson:

Task 4.2: Assignment 1: Interpreting some R commands (5 points)

The sequence of R commands that follows will produce an IDW interpolation of the (*x, y, z*) data in the file `topo_dat.txt` that we used in Lesson 1 but, you will recall, did not proceed to model the surface. All we did was to use color and bubble plots to visualize the basic data. *What I want you to do is to run the same sequence but for every line of the R sequence add an explanation of what that command is doing*. You may well need to use `>help("name of object")` to do this. For example, our first command is interpreted as

```
> rm(list=ls(all=TRUE))
```

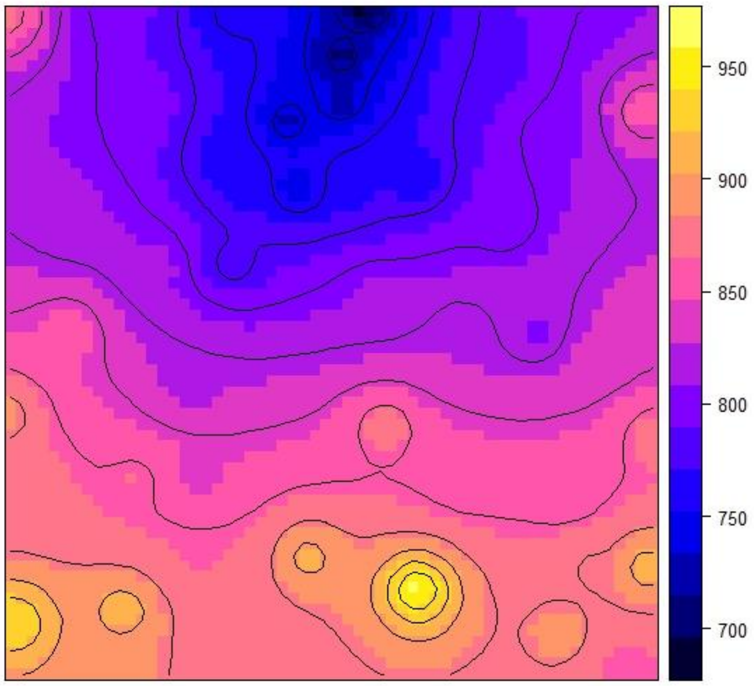
This means: “Create a list of all objects in the (previous) workspace called list and remove it”

ETC, but don't be too verbose and don't just copy the help file accounts!

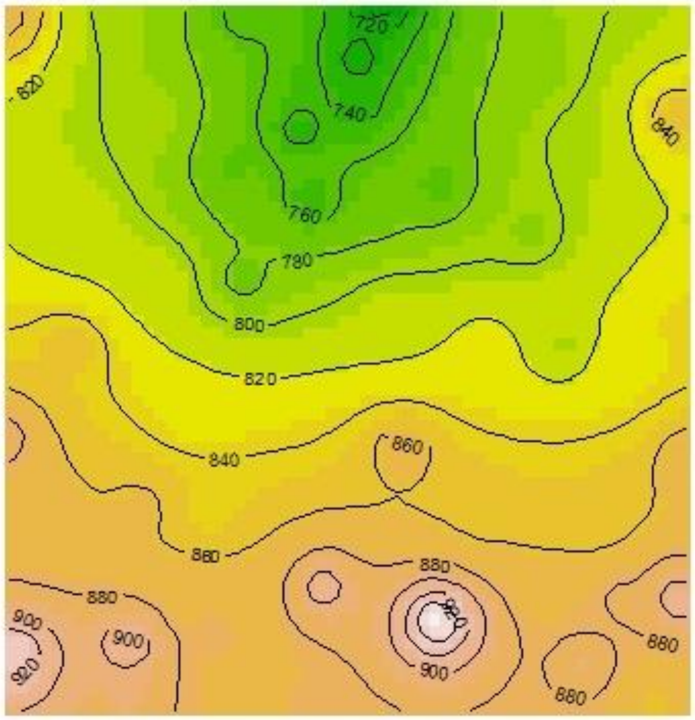
```
> rm(list=ls(all=TRUE))
> topo_datatable <- read.table("C:\\Users\\David\\Desktop\\topo_data.txt", header =
TRUE)
> library(sp)
> coordinates(topo_datatable) <- c("X", "Y")
> topo_grid <- spsample(topo_datatable, "regular", n=3720)
> gridded(topo_grid) <- TRUE
> fullgrid(topo_grid) <- TRUE
> library(gstat)
> topo_idw <- idw(Z~1,topo_datatable,newdata=topo_grid,idp=2.0)
> im <- as.image.SpatialGridDataFrame(topo_idw)
> library(maptools)
> topo_SLDF <- ContourLines2SLDF(contourLines(im))
> topo_spl <- list("sp.lines", topo_SLDF)
> spplot(topo_idw, "var1.pred", sp.layout=topo_spl)
> image(topo_idw, "var1.pred", col=terrain.colors(20))
> contour(topo_idw, "var1.pred", add=TRUE)
>quit()
```

The last command is of course optional!

Don't skip this part of the lesson, it is very important that you follow what we are doing and why. There are several new concepts in the sequence, notably coordinates, spsample with gridded and fullgrid, idw, as.image and SpatialGridDataFrame, ContourLines2SLDF, image and contour. If your perspective comes from a GIS and you are used to a single clicked command to *interpolate*, this step-by-step approach might seem unwieldy, but, as we will see, it is also very flexible and powerful, giving access to a very large number of options for both interpolation and display. It has merit in making every step along the way absolutely explicit. If you run this sequence it will serve as a basic route to IDW and, in fact the entire range of interpolation methods in the R environment. This is because the basic interpolation step will be more-or-less the same every time we use it. The sequence should have produced the two maps shown below:



Topo_data with IDW interpolation, e=2.0, basic output

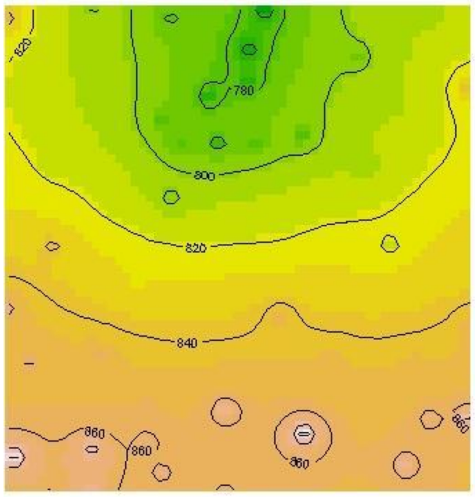


The same, layer colored using a standard palette and with contours labeled

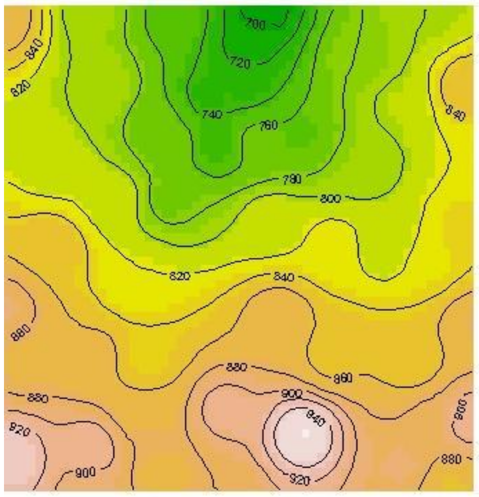
What do you think about the quality of these contour maps? I can see several problems …

IDW interpolation can produce some perfectly reasonable results that do not differ very much from kriged maps when the surface is reasonably smooth but it has a number of defects:

- First, although we have a map of estimates of the surface height, and in common with all deterministic approaches, we have no way of assessing the likely *errors* involved;
- Second, although *leave one out cross validation* can be used (and is available in gstat), in general we have no theoretical reason for the choice of the distance exponent to be used. In the example we used idp=2, which gives a reasonable result and was the standard used in the venerable SYMAP program. SYMAP was developed at Harvard in the mid-1960s and was the first generally used interpolation routine. There is nothing to stop you simply repeating the analysis until you get a map that looks right. I have experimented in this way with IDW and find that in general a distance exponent of 2.0 produces maps that correspond fairly well to those that a human cartographer would usually draw;
- Third, as the basic algorithm is usually coded, only distances from sample control points to the interpolated locations are used and this leads to undesirable *side effects* if the control data are clustered. We might, for example, want points in a cluster to be down weighted in some way;
- Fourth, the dependence on distance can mean that in data sparse areas and with high exponent values, the method produces what in the jargon are called called *ring contours* but most of us would call *bull's eyes*. The '860' contour on the right hand display is a good example but there are others on this map. Fine tuning by changing the distance weighting exponent idp can help here. Setting it at 1.0 exacerbates this problem, whereas setting it 3.0 produces quite a pleasant map:



Topo_data with distance exponent=1.0



Topo_data with distance exponent 3. 0

- Finally, because IDW guarantees that the weights must lie between 0 and 1, interpolated *values can never lie outside the range of the control point data*. In practice there are applications in which a human cartographer would extrapolate beyond the range of the data, for example, putting the top onto a mountain or the bottom into a valley.

These maps also illustrate a further issue that is true for almost all automated interpolation including kriging. We know that in reality this is a topographic surface on which rivers in valleys will always fall downhill, but look at the valley in the center at the 'top' of our left-hand, $e=1.0$ map. Our algorithm has no knowledge that this is earth surface relief and so is happy to produce valleys that, according to the interpolated contours, rise and fall in a way that nature never shows. In production GIS environments there are ways of ensuring that an interpolated digital elevation matrix is consistent with the pattern of river valleys, but this is a luxury we don't have here.

Lastly, please note that IDW is only one of many possible deterministic approaches, which we list and describe on pages 261-262 of the course text and that there are R packages that implement some of them.

4.3 Modeling surface structure

It would obviously be an advance if we provide our interpolation routine with as much information as we can about the spatial structure of the field that we are interpolating. One simple approach, called *trend surface analysis* in the geographical and GIS literature, makes the field height *z* a *specified* function of the general form $z = f(x, y)$ and then fits this to the observed data using ordinary least squares (OLS) regression. In our text we lean towards the term *regression on spatial co-ordinates*, which is exactly what it is, for this approach. Numerous types of function have been experimented with, but in practice nowadays most people seem to use polynomials in (x , y) and, since we often have only very general trends in mind, typically these are of degree 3 at maximum. Section 4.3 (a) explores this approach.

An alternative approach is to let the data speak for themselves and use what can be deduced about the structure of the surface to develop optimum weights for interpolation, which gives a family of approaches that are known as *kriging* (Most people pronounce this as ‘krigging’ but it comes from a South African family name and when I ask my South African friends they make it sound more like ‘kriking’). Section 4.3 (b) explores this.

4.3 (a) Trend surfaces

Task 4.3
Regression on spatial co-ordinates (‘Trend surfaces’)

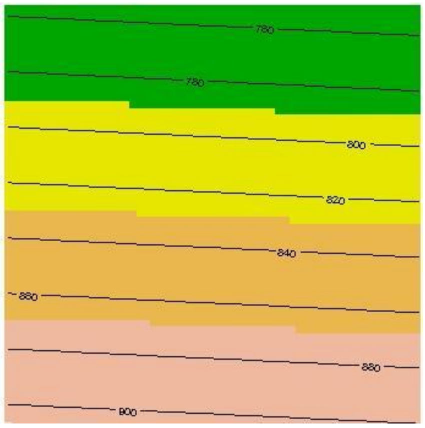
Read Sections 10.1 and 10.2 pages 277-287 of O’Sullivan and Unwin (2010). Please note that there is a hopefully obvious error page 285 where instead of the $df_{\text{residuals}}$ = 10-1-2=7, it should be 24-1-2 = 21.

The same material is Section 9.3, pages 256 - 265 of the first edition and there is a brief reference in Section 8.3.2, pages 194 -195 of Bivand *et al*. (2008). Brunsdon and Comber do not cover the topic.

If you want to take things further there is a very old (but free) student guide *An Introduction to trend surface analysis* that I wrote over 40 years ago and is number 5 in the CATMOG series available at http://qmrg.org.uk/catmog/. (Checked 10-12-15). There are some other great resources in the CATMOG series that you might like to look over later on. Finally, a pdf of a more recent summary account (Unwin, 2009) is available for download from the course website.

We can think of trend surface analysis in two ways. The first is as a global interpolation in which we are happy to accept that our control data, the z height values, have a lot of error and where we are interested simply in any gross, edge to edge, structure in the field we have sampled.   In gstat this approach is recognized by a special version of the krige method as in the following example:

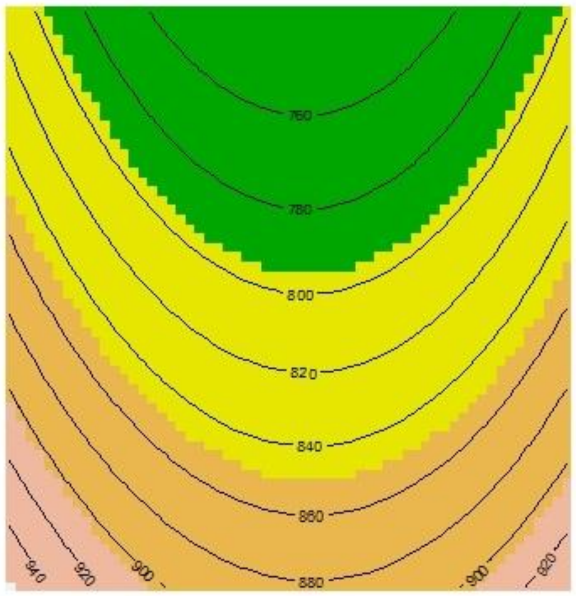
```
> topo_tsa_1 <- krige(Z~1, topo_datatable,
newdata=topo_grid, degree=1)
[ordinary or weighted least squares prediction]
> im <- as.image.SpatialGridDataFrame(topo_tsa_1)
> image(topo_tsa_1,"var1.pred",col=terrain.colors(5))
> topo_tsa_line <-ContourLines2SLDF(contourLines(im))
> contour(topo_tsa_1,"var1.pred", add=TRUE)
```



As will be discovered if you echo the contents of topo_tsa_1, what is placed in the object are the 3720 predicted values (var1.pred) across our 60 x 62 (= 3720) grid. This also means that in a strict sense we do not have a map of the best fit linear surface, but, as the graphic makes obvious, of a pixelated approximation to it. Nor do we have any of the usual fitted values, residuals and goodness of fit information.

A similar sequence yields the quadratic (degree=2) surface:

```
> topo_tsa_2 <- krige(Z~1, topo_datatable, newdata=topo_grid, degree=2)
[ordinary or weighted least squares prediction]
> im <- as.image.SpatialGridDataFrame(topo_tsa_2)
> image(topo_tsa_2,"var1.pred",col=terrain.colors(5))
> topo_tsa_line <-ContourLines2SLDF(contourLines(im))
> contour(topo_tsa_2,"var1.pred", add=TRUE)
```



This is all well and good, but in most applications you will want all the usual regression diagnostics, so it is sensible to use the standard R command lm (linear model). I find it easier, both conceptually and practically, to work directly with a standard data.table and so would re-read the data and work from there as:

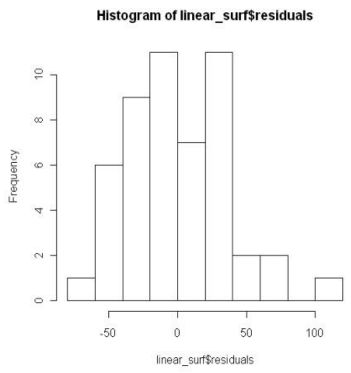
```
> topo_dat <- read. table("C:\\Documents and Settings\\David Un win\\Desktop\\R-
results\\topo_data.txt", header = TRUE)
> x <- topo_dat$X
> y <- topo_dat$Y
> z<-topo_dat$Z
> linear_surf <- lm( z ~ x + y)
```

which gives:

```
> linear_surf
Call:
lm(formula = z ~ x + y)
Coefficients:
(Intercept)           x           y
   910.660       -1.170      -24.649
> quad_surf <- lm(z ~ x + y + I(x*x)+I(y*y) + I(x*y))
> quad_surf
Call:
lm(formula = z ~ x + y + I(x * x) + I(y * y) + I(x * y))
Coefficients:
(Intercept)          x               y         (x * x)            I(y * y)            I(x * y)
  978.0298     -53.4986     -30.5797     7.4792           .0474              0.2745
```

The class lm objects linear_surf and quad_surf contain various results (use >help("lm") to list them) which can be accessed for further use, for example:

```
> hist(linear_surf$residuals)
```



It is also useful to list the various goodness of ft and significance measures using:

```
>summary(linear_surf)
```

which gives:

```
Call:
lm(formula = z ~ x + y)

Residuals:
   Min          1Q          Median       3Q          Max
-63.623       -25.984       -2.748       22.765      110.051

Coefficients:
              Estimate      Std. Error   t value     Pr(>|t|)
(Intercept)    910.660      13.593       66.993      < 2e-16 ***
x             -1.170        2.828        -0.414       0.681
y             -24.649        2.660       -9.266      3.53e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 36.84 on 47 degrees of freedom
Multiple R-squared: 0.6472,     Adjusted R-squared: 0.6322
F-statistic: 43.11 on 2 and 47 DF,  p-value: 2.331e-11
```

This linear fit is both strong and highly significant, but note that it is the variation in along the Y-axis that is important, the trend is almost exactly in the north/south direction.   Also note for reference when we come to interpolate these data using kriging that there is therefore a very substantial *drift* (aka *trend*) in the mean surface height.

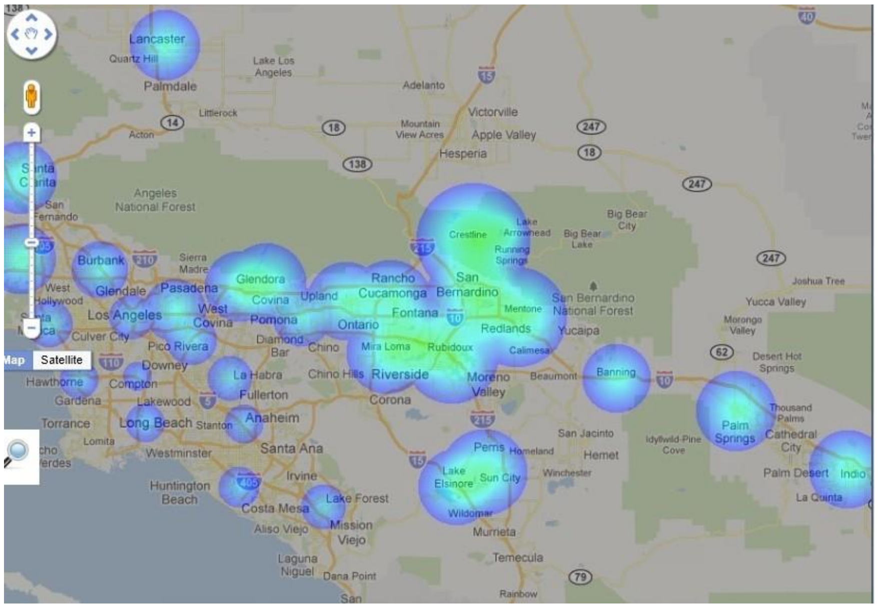
Viewed as a formal model fitting exercise, I think it fair to emphasize the view (page 287 of the course text) that the theoretical underpinnings of trend surface analysis are weak and that almost always autocorrelation amongst the residuals will indicate that the model is mis-specified, such that many statisticians have little time for the entire approach.

However, you can be too purist about these things.   The approach is nowadays not as fashionable as it was in the 1970s and 1980s, but, whatever very formal statistical theory might have to say on the matter, there are hundreds of practical studies that seem to have found it useful and, as we will see, it is often used in universal kriging to model any drift in mean *z* across a surface. My CATMOG, referenced above, provides an early introduction to some of the issues in surface fitting of this sort, and there is an update (Unwin, 2009) available *via* our course learning management system.

Task 4.4

Assignment 2 Fitting trend surfaces in R (10 points)

Your task for this second part of the assignment for this lesson is to use the tools in R to fit linear and quadratic polynomials to the data file (Simple_LA_OZ.txt that we used in Lesson 1 and that can be downloaded from the learning management system. Show the code you used and the resulting displays, but also comment on what the results seem to indicate. For example, is the quadratic surface an improvement on the simple linear one and, if so, why/why not? Are there any really troublesome points? I have located these data onto WGS84 in Google Earth™ and the result may well be useful for you when interpreting the resulting surfaces:



4.3 b) Variography

Task 4.5

The Square Root Differences Cloud and the (Semi)Variogram

Now read Section 10.3, pages 287-302 of O'Sullivan and Unwin (2010) on the square root difference cloud, sample variogram, and methods for modeling the latter (The 2002 First Edition has more-or-less the same materials in Section 2.4, pages 45 - 49 and 266 - 274).

It will help your understanding of this section if you also read and have by your side the R help documentation on variogram and fit.variogram, both in the gstat package, using:

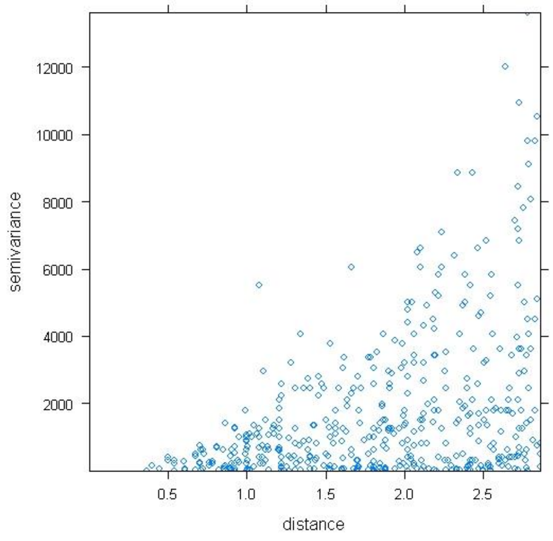
```
>library (gstat)
>??variogram
>??fit.variogram
```

Brunsdon and Comber (2015) have a brief introduction to this approach in Section 6.8 pages 208-216. Bivand *et al* (2008) Section 8.4, pages 195 - 209, introduce some of the complexities of variogram formulation and model estimation in the R environment

Note that in the course text we follow the advice of Cressie (1993, page 41-42) and, for the reasons given, initially show the *square root difference* cloud rather than the sample variogram. Often people don't even bother to show the variogram cloud and go straight to the summarizing sample variogram. My view is that showing the cloud is useful in that it might help pick up any erroneous data pairs and there is a lot that can be done at this stage to help our understanding of the surface structure. First, we generate and plot a straightforward variogram cloud:

```
> topo_vg <- variogram(Z~1, data= topo_datatable, cloud=TRUE)
> plot (topo_vg)
```

This gives the display below which, with the exception of the shorter cut off used (see below), is identical to Figure 10.7, page 291 of the course text:



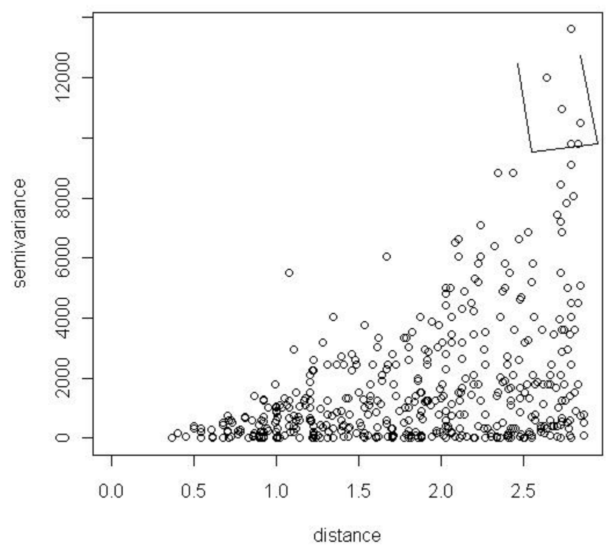
There is a neat little trick that enables you to see where possibly-aberrant pairs of values occur on the map. First, compute and plot the variogram cloud using (take care when typing this nested set of objects!):

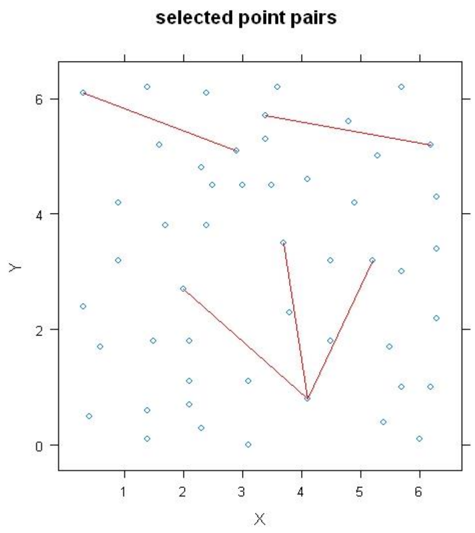
```
> topoSV_edit <- plot(variogram(Z~1,topo_datatable,cloud=TRUE),digitize=TRUE)
[1] "mouse-left digitizes, mouse-right closes polygon"
```

This will invite you to use the mouse to select an area of the cloud for analysis using the left button to digitize (it's likely to be a little bit clunky!) and then close the polygon using a right mouse button. Using plot will now identify the point pairs that lay within the selected area of the graph:

```
> plot(topoSV_edit,topo_datatable)
```

The two displays below show the points I wanted to examine and the location of the pairs of points involved. In this case maybe I'd start to worry a little about the large variance in the south to north direction. In fact, almost all the very high variance pairs are pairs that are separated by a large distance in this direction.



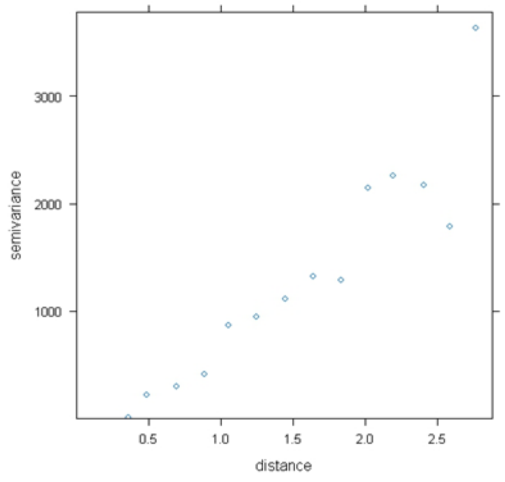


Selecting five extreme point pairs *Displaying them*

The summary variogram is returned by default when we omit the cloud=TRUE argument:

```
> topo_vg <- variogram(Z~1, data= topo_datatable)
> plot(topo_vg)
```

This gives the figure below, which can be compared with our graphic, Figure 10.8, page 292:



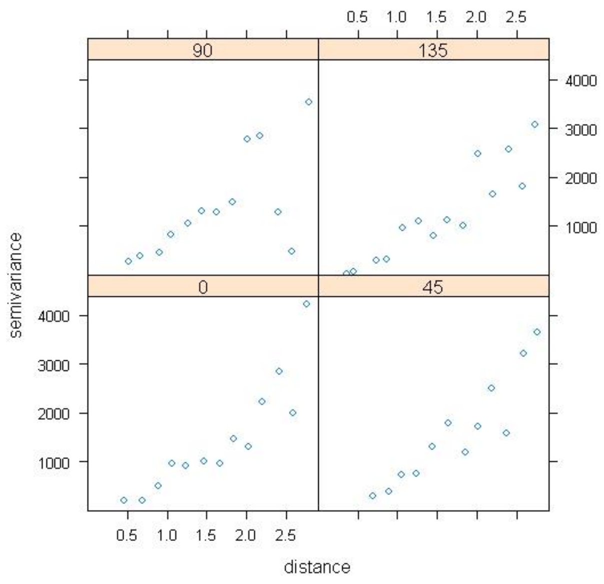
I would be the first to admit that, although fairly typical, this sample variogram is unlikely to be easy to model it is probably better to widen the bins to provide a smoother curve. Access to the quantities contained in the variogram object is straightforward:

```
> topo_vg
        np      dist            gamma           dir.hor         dir.ver         id
1       1       0.3605551       4.5000          0               0               var1
2       10      0.4862763       221.8000        0               0               var1
3       16      0.6897883       298.3125        0               0               var1
4       28      0.8844888       414.0000        0                0              var1
5       36      1.0556110       862.5000        0               0               var1
6       41      1.2481436       944.0488        0               0               var1
7       31      1.4496666       1114.0000       0               0               var1
8       34      1.6410473       318.4853        0               0               var1
9       44      1.8352664       1289.7159       0               0               var1
10      43      2.0191807       2141.7558       0               0               var1
11      41      2.1920599       2259.4268       0               0               var1
12      46      2.4045231       2174.9783       0               0               var1
13      42      2.5846755       1785.3095       0               0               var1
14      51      2.7687799       3636.8922       0               0               var1
```

By default variogram makes a few decisions for you, using 1/3 of the length of the longest diagonal of the bounding box as its *cut off* value beyond which no point pairs are considered.   This is to avoid introducing pairs of values that are so distant they do not help in our understanding of the surface structure. The argument cutoff lets you change this. Similarly, variogram also chooses 15 bins over this distance in which to summarize the variogram cloud. The argument width enables you to change this.

Clearly (as above?) narrow bin widths will introduce detail that may or may not prove useful. Finally, by default variogram computes the variances for all directions but it is sometimes useful to examine the variation stratified by direction using the argument alpha as for example:

```
> plot(variogram(Z~1,topo_datatable, alpha = c(0,45,90,135)))
```



Hopefully, you can see that, there is a lot more to what's called *variography* than simply allowing your GIS to default to some values, however sensible these might be, but hopefully you can also see that different spatial dependencies across a geostatistical surface will generate different sample variograms. It is worth noting that it is extremely unlikely that any real–world generating spatial process is *completely* uncorrelated so that in practice almost always there will be some structure to be uncovered. The challenge is to explore the data using these tools and find it.

The most common use for the structural information contained in the variogram is for spatial interpolation by kriging and it to this that we now turn.

Task 4.6

The mathematics of interpolation (spatial prediction) by Kriging

Now study O'Sullivan and Unwin (2010), pages 302 – 310 on using a modeled variogram to produce optimum weights by ordinary kriging. In the First Edition, much the same material is on pages 274 – 281.

Bivand *et al.* (2008) cover the same materials at much more advanced level in pages 201 *et seq*.

Our sample variogram has values for the variance at the mid-points of the distance bins used and clearly the values we have are simple estimates of some underlying true values, yet the true variogram is continuous and it is this that we really would want to know. It might be thought that almost any closely fitting curve will provide estimates of the variances at all distances, yet for the kriging equations to be soluble we cannot allow functions that will give rise to negative variances from combinations of random variables. Only functions that ensure positive non-zero variances can be used to model the sample variogram. In the literature these are called *authorized* models.

A list of the variogram functions available in gstat can be found from:

```
>vgm()

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

This is a long list, but in practice not all are equally useful. Most practical studies use one or other of the *linear*, *spherical* or *Gaussian*. The characteristics of these models are described and illustrated in most standard texts, but one I can recommend is the treatment by Webster and Oliver (2001, Chapter 6, pages 105 -134). Individual models are specified using vgm as:

```
> vgm(1,"Sph", range=390)
```

```
  model psill range
1   Sph     1   390
> vgm(0,"Lin")
  model psill range
1   Lin     0     0
```

So how do we arrive at and fit a variogram function (or functions, since we can mix functions if it seems necessary) to our sample variogram to generate what in the literature is called the *experimental variogram*? There are several approaches to what has been called the *black art* of variogram modeling:

- Select as robust a model as possible (typically the spherical) and fit it to the sample variogram using standard methods. This is the approach adopted by some GIS and it is not as silly as it might seem since in practice 'even a fairly crudely determined set of weights can give excellent results when applied to data' (Chiles and Delfiner, 1999, page 175);
- Experiment with a series of models looking at the fitted curves and using some goodness of fit measure until you arrive at one that seems satisfactory. Historically the computer program called *Variowin* (Pannatier, 1995) has been used a great deal to facilitate this approach. Note that fitting uses iteration and that some starting values are required, which means that care has to be taken to ensure that a fit will actually succeed. Note also that gstat allows partial fitting for circumstances where we know (or think we know) the true value of a parameter and wish to estimate the remainder;
- Do the above but automatically (see below);
- Simple visual fitting, either 'by eye' using pencil and paper or via a computer display, but then converting the result into an authorized model. The R package geoR provides a GUI and conversion function that enable this approach and many practicing users are happy to work in this way. It is not as 'silly' as you might think!

The solution that commends itself is to apply Occam's razor and fit a simple authorized function that seems to make sense. You can spend hours and hours trying to improve a model for negligible gain in fit and validity of the resulting interpolation.

```
> top_v_fit <- fit.variogram(topo_vg, vgm(2000, model="Sph", range=2.5, nugget=0.0))
```

I have experimented with numerous variogram models for these data. This one seems to work, but you might equally well fit a bounded linear or, given the way the experimental variogram approaches the origin, perhaps a Gaussian?

A note on automation

There is an automated approach developed by Paul Hiemstra and available in his automap package. In this the method autoKrige peforms automatic Kriging on the given dataset and the experimental variogram is generated automatically using autofitVariogram. Here is what it makes of our topo_data:

```
> d<-read.table("C:\\Users\\David\\Desktop\\topo_data.txt",header=T)
> attach(d)
> library(sp)
> coordinates(d)=~X+Y
> library(automap)
> krige=autoKrige(Z~1,d)
[using ordinary kriging]
```

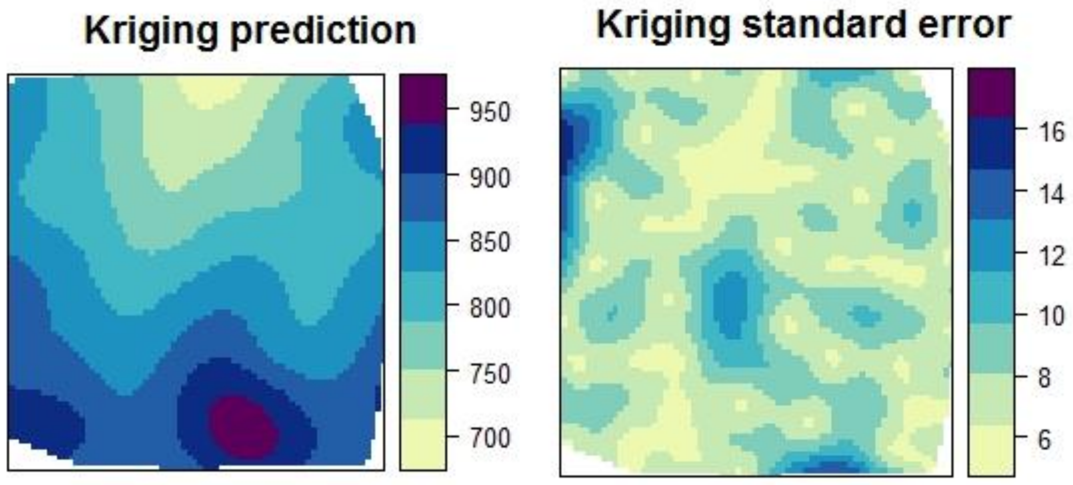
AutoKrige returns an object (krige) containing the results of the interpolation (predicted value, variance and standard deviation) over either a user specified or a default area (This is about 5000 points in the convex hull of the data points), the sample variogram, the variogram model that was selected and fitted and the sums of squares between the sample variogram and the fitted variogram model. These are called krige_output, exp_var,var-model and sserr respectiviely:

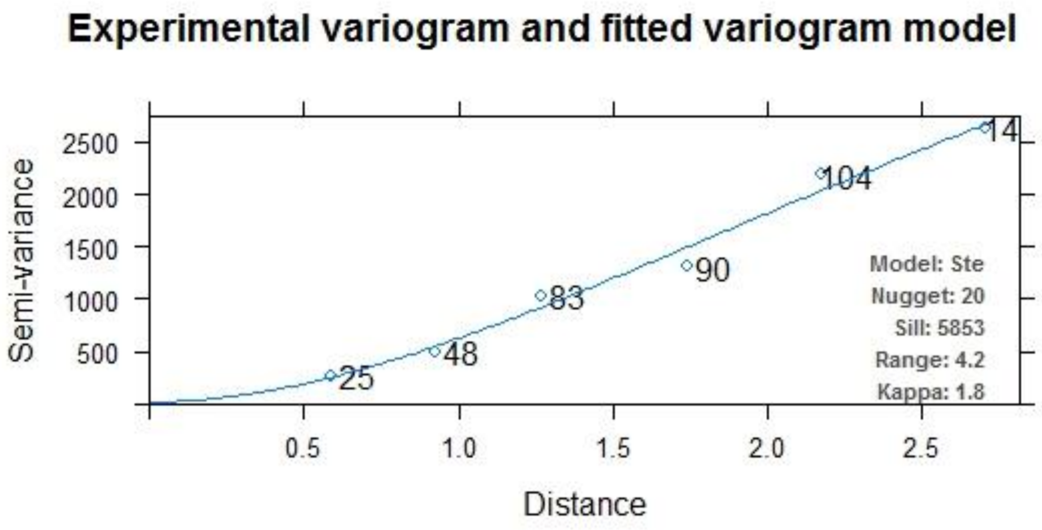
```
krige$exp_var
   np      dist     gamma dir.hor dir.ver   id
1  25 0.5894711  264.3600       0       0 var1
2  48 0.9259567  493.6146       0       0 var1
3  83 1.2684345 1033.9819       0       0 var1
4  90 1.7431937 1305.5000       0       0 var1
5 104 2.1754214 2189.6923       0       0 var1
6 145 2.7062943 2638.9793       0       0 var1

krige$var_model
  model     psill    range kappa
1   Nug   20.2356 0.000000   0.0
2   Ste 5832.9557 4.168967   1.8

krige$sserr
[1] 2464018

> plot(krige)
```





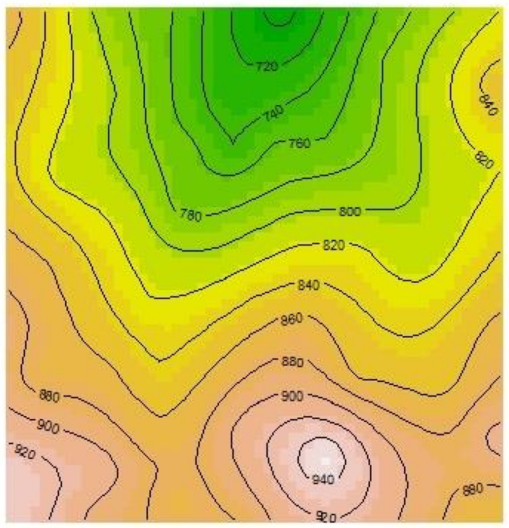
You will note (see below) that I select a different model and cartography but the resulting maps don't differ much. Some years ago I chaired a webinar at which Dr Hiemstra presented automap and it is important to understand that the context was in developing an automated warning system based on near real-time monitoring of an environmental pollutant applicable to locations at which the pollutant was not measured. In such an application, there is a need for automation and for error estimates. I am not convinced that automating the structural analysis is always good practice but I have to admit that in the (few) cases for which I have checked a manual analysis using automap the results have been impressive.

4.4 Interpolation by Kriging

Assuming that we still have our original topo_datatable, a grid (topo_grid) to receive the interpolated values, a fitted variogram model, in this case a spherical one contained in top_v_fit, and have the relevant libraries (sp, maptools and gstat) loaded, producing a krige interpolation is easy, if , as with IDW interpolation, a little tedious:

```
> topo_map_OK <- krige(Z~1, topo_datatable, topo_grid, top_v_fit)
> topo_im <- as.image.SpatialGridDataFrame(topo_map_OK)
> topo_SLDF <- ContourLines2SLDF(contourLines(topo_im))
> topo_sp1 <- list("sp.lines", topo_SLDF)
> image(topo_map_OK,"var1.pred",col=terrain.colors(20))
> contour(topo_map_OK,"var1.pred", add=TRUE)
```

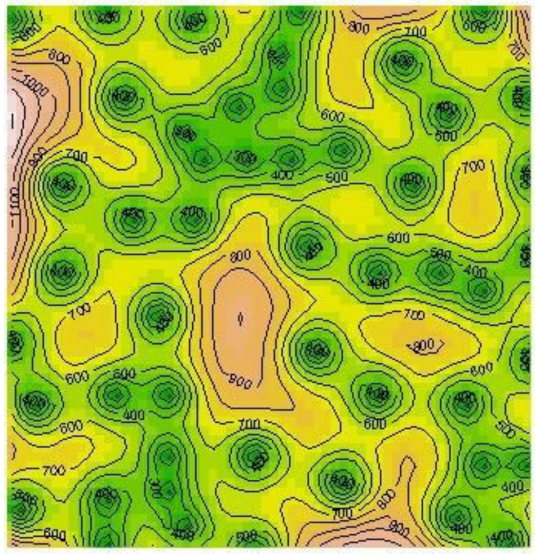
Note that only the first line of this is necessary to calculate the estimates, the rest are to create a display and are a sequence you have seen before.



My view is that this is a lot better than our original IDW and it has the supreme advantage of having also available, as the variable var1.var in the topo_map_OK object, variance estimates that can also be mapped:

```
>image(topo_map_OK,"var1.var",col = terrain.colors(20))
> contour(topo_map_OK,"var1.var", add=TRUE)
```

The influence of proximity to a data point is fairly obvious here, but this type of map could be used, for example, to suggest where additional data are or are not required:



Task 4.7

Other members of the Kriging family

Now read pages 310-312 of O'Sullivan and Unwin (2010), which is an expansion of material on page 281 of the First Edition but which cannot do more than scratch the surface of alternative forms of kriging, each of which in essence reflects a different set of assumptions made about the surface/field we have sampled.

Bivand *et al*. pages 209-231 extend this a lot further, but I suspect that to follow what they illustrate using R, you will also need to do some further reading in the literature of geostatistics.

All are implemented in the krige prediction method in gstat, with the software actually making the decision as to which to use according to the information supplied in the arguments you supply. If no variogram is supplied then the method defaults to *inverse distance weighted interpolation* using idw. If we are willing to specify a mean (β) as a single value, the default is *ordinary kriging*, but if we provide it as a vector of trend surface coefficients the default is *simple kriging*, Provide neither and you get *universal kriging*.

Task 4.8

Assignment 3: Variography and interpolation by Kriging (10 Points)

The final task is simple, but likely to take an hour or so to do. Using the Simple_LA_OZ.txt data from Task 4.4 and Lesson 1, produce (a) the variogram cloud, (b) a sample variogram (c) a suitable model of this and (d) a map of the krige estimates and comment briefly on these results.

STRONG HINT: you should consider whether or not you wish to delete data point 31 either before or during your analysis. You will perhaps recall that visualization using an IDW interpolation shows it to be locally very anomalous.

4.6 Conclusion

Well, that's far enough for now. When I started to prepare these four lessons I had an ambitious plan that would have taken us much further into the world of geostatistics, for example to examine *multivariate variogram* modeling and prediction, *co-kriging* where we use information from some co-variate to improve spatial predictions, issues of changes in the 'support' that are dealt with in *block kriging*, cross validation of the results, and the rapidly expanding fields of *geostatistical simulation* and *model based geostatistics*. All of this is available in gstat. This plan had to be revised and for two reasons. First, I reckoned without the richness of the R environment for almost all spatial analysis and, second, I realized that a complete cover of the field would need many more (and more difficult) weekly lessons.

In no sense have we covered everything that is available in R for even the limited geostatistical problems we have examined this week. There are other R packages could have been used. For deterministic interpolation you might install and look at the package fields for its Tps (thin plate smoothing) function or akima and/or stinepack. For geostatistical interpolation I think that the package spatial will replicate all the analyses we have done and RandomFields covers geostatistical simulation. For more general geostatistics, both geoR and fields contain functions for variography and kriging.

Hopefully, this 'orientation' will both persuade and enable you to explore further.

4.7 Data files used

topo_data.txt : from s.com website, modified NOTREDAME teaching data from J.C. Davis (2002) *Statistics and Data Analysis in Geology* (NY: Wiley) (with John's kind permission)

Simple_LA_OZ.txt: modified to a unit square and into z-scores from a file downloaded from the *geodatacenter @ asu*

4.8 References

Chiles, J-P and P. Delfiner (1999) *Geostatistics: Modeling Spatial Uncertainty* (Chichester & NY: Wiley)

Clark, Isobel (2001) *Practical geostatistics*, 119 pages available at (checked 11-12-15)

Davis, J.C. (2002) *Statistics and Data Analysis in Geology* (Third Edition, Chichester & NY: Wiley)

Isaaks, E.H. and R.M. Srivastava (1989) *An Introduction to Applied Geostatistics* (Oxford: Oxford University Press)

Pannatier, Y. (1995) *Variowin: Software of Spatial Analysis in 2D* (New York: Springer Verlag). See www.aigeostatistics.org for up to date information on this and other similar programs.

Webster, R. and M.A. Oliver (2001) *Geostatistics for Environmental Scientists* (Chichester & NY: Wiley)

Unwin, D. (undated, but 1975) *An Introduction to Trend Surface Analysis*, Concepts and Techniques in Modern Geography (CATMOG), 5, available as pdf from www.qmsg.org.uk/catmog, 40 pages

Unwin, D. J. (2009) Trend surface models. In: Kitchin, R. and N. Thrift (eds.) *International Encyclopedia of Human Geogra*phy, Volume 11, pp. 484-488 (Oxford: Elsevier). (Reprint of © Elsevier material available on the learning management system by courtesy of the publisher)