Applied geostatistics
Exercise 8: Spatial sampling

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

June 26, 2010

Contents

1 Introduction **1**

2 Design-based sampling **3**
2.1 Completely random spatial sampling 3
2.2 Evaluation: reproducing population statistics 4
2.3 Evaluation: reproducing the variogram 7
2.4 Answers 9

3 Grid sampling **10**
3.1 Grid sampling with jitter 11
3.2 Removing points outside the sample area 12
3.3 * Overlaying with a bounding box 13
3.4 Evaluation: reproducing population statistics 17
3.5 Evaluation: reproducing the variogram 18
3.6 Answers 19

4 Optimal grid sampling **20**
4.1 Answers 22

5 Self-test **22**

References **25**

Version 1.91 Copyright © 2007–2010 University of Twente, Faculty ITC All rights reserved. Reproduction and dissemination of the work as a whole (not parts) freely permitted if this original copyright notice is included. Sale or placement on a web site where payment must be made to access this document is strictly prohibited.
To adapt or translate please contact the author (http://www.itc.nl/personal/rossiter).

Index of R concepts **26**

顺藤－摸瓜 (Shù téng – mōguā)
"Follow the vine to grasp the melon"
– Chinese folk saying

1 Introduction

Our knowledge of nature comes from **samples**, that is, a set of **sampling units** (sometimes loosely called "samples") taken from the **sampling frame**, which is all possible units in the underlying **population**. On the basis of the sample we make **inferences** about the population; e.g. we predict on a grid of many thousands of locations based on a sample of a few hundred observations.

Sampling is expensive and time-consuming, so designing a good sampling scheme (maximum information at minimum cost) is an important application of geostatistics.

After completing this exercise you should be able to:

1. Design completely random, stratified random, and grid sampling schemes;
2. Find the "optimal" regular grid sample for a given variogram model.

A general reference relevant to this exercise is de Gruijter et al. [4]. Also of interest is a US Forest Service technical report [12] and articles by Stein and Ettema [13].

For sampling to determine the variogram, relevant papers are by Webster et al. [16], Lark [6], Müller and Zimmerman [10], Warrick and Myers [15], Russo [11], and van Groenigen et al. [14].

Sampling for kriging with external drift has been recently addressed by Hengl et al. [5], Minasny et al. [9] and Brus and Heuvelink [2].

In this exercise we use the Jura soil sample area, i.e. the bounding box of the prediction grid `jura.grid` introduced in Exercise 4. We will also use a simulated field, as in Exercise 6 §5, to evaluate the success of the various sampling schemes; this will be recreated from the calibration points `jura.cal` and the fitted variogram model `vmf`.

Task 1 : Load the saved dataset from Exercise 4 §5. This should include:

1. Calibration points `jura.cal`
2. Validation points `jura.val`
3. Prediction grid `jura.grid`

as spatial objects. •

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

We will make one simulated field and pretends that it is a perfect map of the population. We can then see how well the various sampling schemes perform.

Task 2 : Compute and plot a single simulated field from the calibration points `jura.cal` and the fitted variogram model `vmf`. •

For a more accurate reproduction of the variogram than in Exercise 5 §6, we use a larger number of points (parameter `nmax`). So that we are looking at the same field, we set the random number generator to the same starting point, with the `set.seed` method. The argument is an arbitrary integer; as long as we all use the same one, we will get the same results.

First we set the seed:

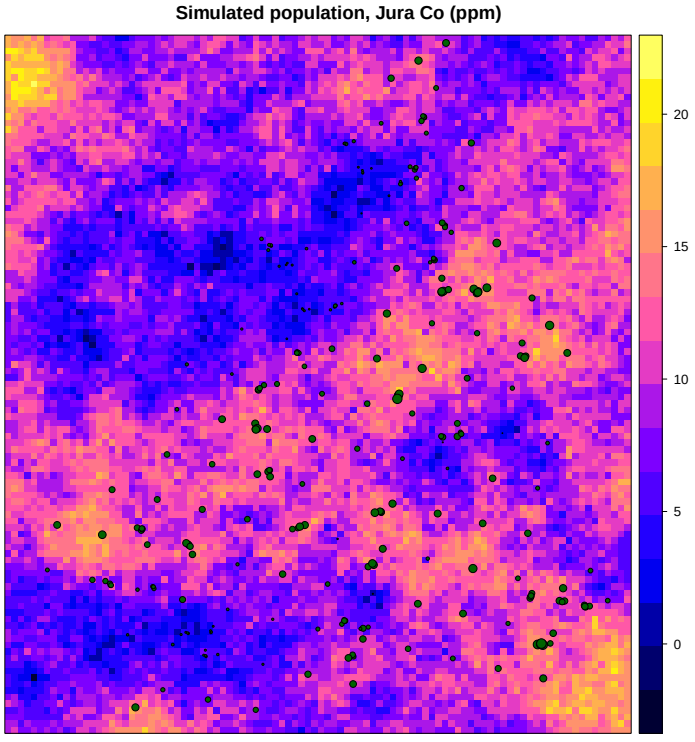
```
> set.seed(619621)
```

Then we compute the simulation object. **Warning!** this will take several minutes (or more on a slower machine).

```
> k.sim <- krige(Co ~ 1, loc = jura.cal, newdata = jura.grid,
+     model = vmf, nsim = 1, nmax = 256)
```

> **Note:** We display the sample points overlaid on the simulated field by means of the `sp.layout` optional argument to the `spplot` method. This takes a list of elements to be plotted, in this case points. Using `sp.layout` is an alternative to using panel functions.

```
> layout.2 <- list("sp.points", jura.cal, pch = 21, cex = 1.2 *
+     jura.cal$Co/max(jura.cal$Co), col = "black", fill = "darkgreen")
> print(spplot(k.sim, zcol = 1, col.regions = bpy.colors(64),
+     main = "Simulated population, Jura Co (ppm)", sp.layout = list(layout.2)))
> rm(layout.2)
```



We will evaluate various designs against this "known" reality:

1. How well can the sample estimate the mean and variance?
2. How well can the sample be used to develop a model of spatial dependence (variogram)? Note that we know the true variogram; it was used to simulate the field.

For reference, the mean and variance of the simulated field are:

```
> mean(k.sim$sim1)

[1] 9.28

> var(k.sim$sim1)

[1] 11.225
```

2 Design-based sampling

Design-based sampling schemes refer to those where each sampling unit has a known probability of being selected. Analysis can then ignore the spatial distribution and use classical (non-spatial) inference. These have been shown to give correct inferences [1], but are not optimal if there is any spatial structure. These designs are extensively discussed in texts such as that by Cochran [3].

2.1 Completely random spatial sampling

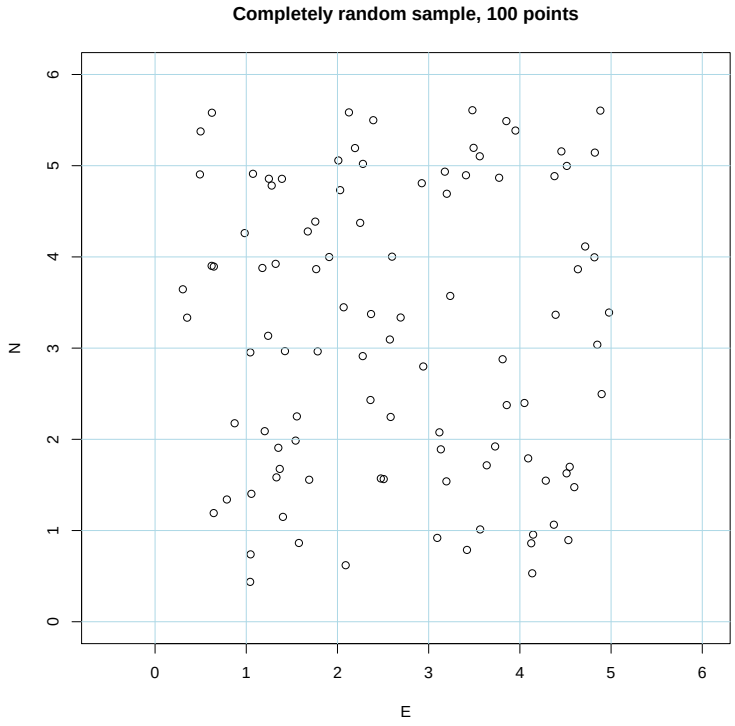
We start with a **completely random** design, where each location has an **equal probability** of being included.

Task 3 : Place 100 sample points randomly over the prediction grid's bounding box; plot their locations. •

It is easy to build the sample from two uniform distributions, one for each axis; but the `spsample` method of the `sp` package already draws a sample in as a spatial object, of class `SpatialPoints`. To get random locations, we specify the `type` argument as `"random"`. We extract the coördinates for the scatterplot with the `coordinates` method:

```
> samp.rand <- spsample(jura.grid, n = 100, type = "random")
> str(samp.rand)

Formal class 'SpatialPoints' [package "sp"] with 3 slots
  ..@ coords     : num [1:100, 1:2] 1.21 2.71 3.19 3.76 4.76 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "s1" "s2"
  ..@ bbox       : num [1:2, 1:2] 0.302 0.424 4.987 5.706
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "s1" "s2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

```
> plot(coordinates(samp.rand), asp = 1, xlab = "E", ylab = "N",
+     xlim = c(0, 5.5), ylim = c(0, 6), main = "Completely random sample, 100 points")
> grid(lty = 1, col = "lightblue")
```



Q1 : *Describe the spatial coverage of this sample. Is this relevant, given the assumptions behind design-based sampling?* *Jump to A1* •

2.2 Evaluation: reproducing population statistics

There are several ways to evaluate the quality of a sampling scheme, if we have a known population to compare it to, as in this case. The simplest is to see how well the sample reproduces population statistics.

The first step is to find the attribute values at the sample points. Of course, in reality we would have to go the field; but here we have an assumed complete population (the simulated field).

Task 4 : Extract the simulated Co values at the sample points. •

The sample points do not usually fall directly on the centre of a grid cell; they can be anywhere in the grid's bounding box, as we can see by comaring the coördinates of the first point (extracted with the `coordinates` method):

```
> coordinates(samp.rand)[1, ]
```

```
     s1      s2
1.04235 0.43743
```

```
> coordinates(k.sim)[1, ]

   s1    s2
0.325 5.725
```

Fortunately, the `sp` package provides an `overlay` method that, among many other things, can overlay an object of class `SpatialPixelsDataFrame` (here, the random field `k.sim`) and an object of class `SpatialPoints` (here, the random sample `samp.rand`). The overlay returns the grid cell number where each point of the random sample falls:

```
> ov <- overlay(k.sim, samp.rand)
> str(ov)

 int [1:100] 10191 1218 1654 2798 8012 5011 6650 8855 3582 5295 ...
```

We can then use these row numbers to extract the data values:

```
> Co.rand <- k.sim$sim1[ov]
```

Q2 : *How well does this sample reproduce the population mean, variance, extremes, and quartiles?* *Jump to A2* •

```
> summary(Co.rand)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -0.45    7.61    9.81    9.50   11.40   18.70

> summary(k.sim$sim1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -1.76    6.91    9.28    9.28   11.60   21.40

> var(Co.rand)

[1] 10.833

> var(k.sim$sim1)

[1] 11.225
```

Task 5 : Display the distribution of the sample, alongside that of the full distribution, as a side-by-side frequency histogram. •

We use the `hist` method to determine the histogram frequencies for specified bins; then we join these into a data structure that can be used by the `barplot` method to produce the side-by-side plot.

```
> hist.rand <- hist(Co.rand, breaks = seq(-4, 24, by = 2),
+     plot = F)
> str(hist.rand)

List of 7
 $ breaks     : num [1:15] -4 -2 0 2 4 6 8 10 12 14 ...
 $ counts     : int [1:14] 0 1 0 5 12 10 26 23 16 5 ...
 $ intensities: num [1:14] 0 0.005 0 0.025 0.06 0.05 0.13 0.115 0.08 0.025 ...
```

```
 $ density    : num [1:14] 0 0.005 0 0.025 0.06 0.05 0.13 0.115 0.08 0.025 ...
 $ mids       : num [1:14] -3 -1 1 3 5 7 9 11 13 15 ...
 $ xname      : chr "Co.rand"
 $ equidist   : logi TRUE
 - attr(*, "class")= chr "histogram"
```

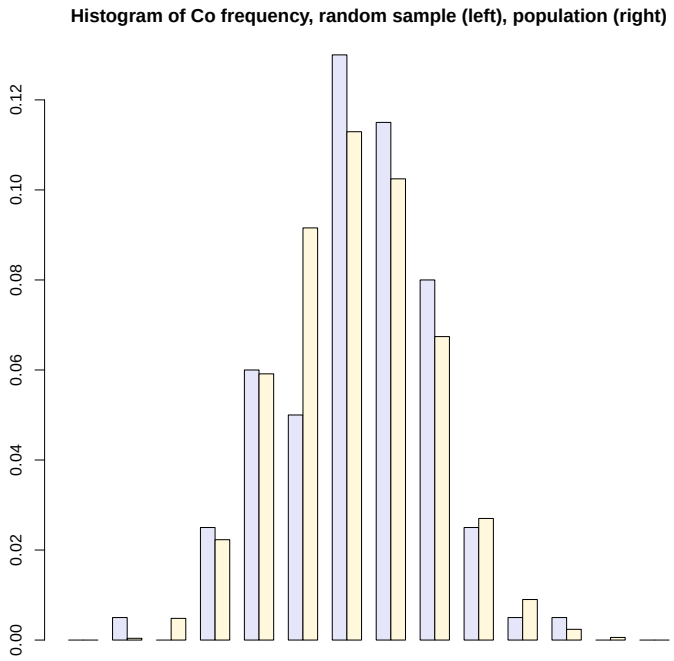
```
> hist.sim <- hist(k.sim$sim1, breaks = seq(-4, 24, by = 2),
+     plot = F)
> str(hist.sim)

List of 7
 $ breaks     : num [1:15] -4 -2 0 2 4 6 8 10 12 14 ...
 $ counts     : int [1:14] 0 8 99 458 1215 1881 2320 2105 1385 555 ...
 $ intensities: num [1:14] 0 0.000389 0.004819 0.022294 0.059141 ...
 $ density    : num [1:14] 0 0.000389 0.004819 0.022294 0.059141 ...
 $ mids       : num [1:14] -3 -1 1 3 5 7 9 11 13 15 ...
 $ xname      : chr "k.sim$sim1"
 $ equidist   : logi TRUE
 - attr(*, "class")= chr "histogram"
```

Note that the structure is the same: both histogram objects have the same number of midpoints and densities, and the midpoint values are the same. So we make one structure:

```
> to.plot <- rbind(samp = hist.rand$density, pop = hist.sim$density)
```

Q3 : *Why do we use the histogram* ***density*** *(proportion of the total) rather than the histogram* ***frequency*** *(actual counts)?* *Jump to A3* •

```
> barplot(to.plot, beside = T, col = c("lavender", "cornsilk"),
+     main = "Histogram of Co frequency, random sample (left), population (right)")
```



Q4 : *Which histogram is more regular? Why?* *Jump to A4* •

Clean up:

```
> rm(hist.rand, hist.sim, to.plot)
```

2.3 Evaluation: reproducing the variogram

We know there is strong spatial structure in this field; could we have discovered this from this sample?

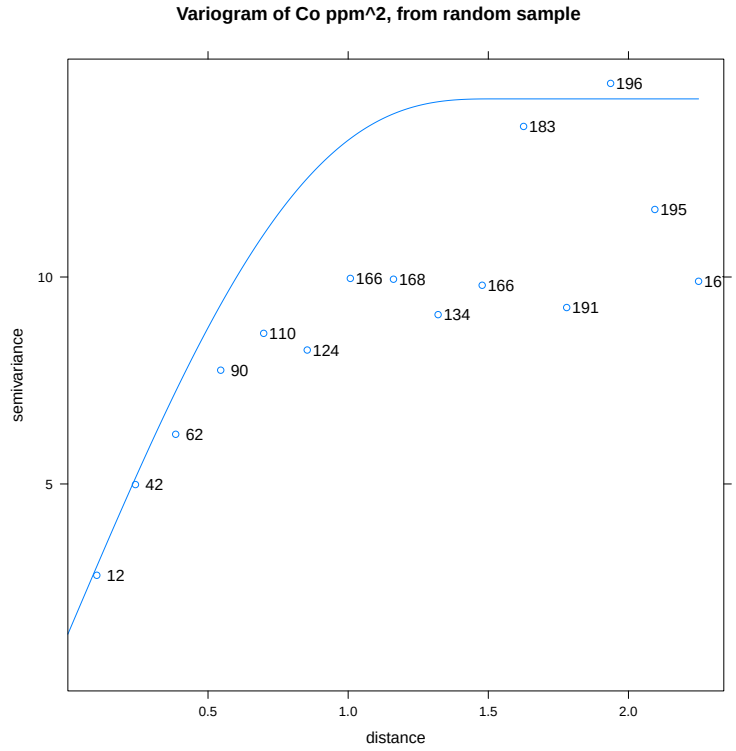
Task 6 : Compute the experimental variogram for Co at the sample points. Plot it with the known model. •

We first make a data frame of just the Co values, with the `data.frame` method, and then add coördinates to it with the `coordinates` method. Note that we use `coordinates` on both the right side (to extract from `samp.rand` and left side (to add to `Co.rand`).

```
> Co.rand <- data.frame(Co = Co.rand)
> str(Co.rand)

'data.frame':        100 obs. of  1 variable:
 $ Co: num  11.24 9.45 10.92 3.24 11.07 ...

> coordinates(Co.rand) <- coordinates(samp.rand)
> str(Co.rand)
```

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        100 obs. of  1 variable:
  .. ..$ Co: num [1:100] 11.24 9.45 10.92 3.24 11.07 ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:100, 1:2] 1.042 3.56 1.391 0.982 2.476 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "s1" "s2"
  ..@ bbox       : num [1:2, 1:2] 0.304 0.437 4.977 5.608
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "s1" "s2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

```
> v <- variogram(Co ~ 1, loc = Co.rand)
> print(plot(v, pl = T, model = vmf, main = "Variogram of Co ppm^2, from random sample"))
```



Q5 : *How well is the experimental variogram computed from the sample fitted by the known model? Would another model form appear more appropriate? Can this experimental variogram be modelled?* *Jump to A5*

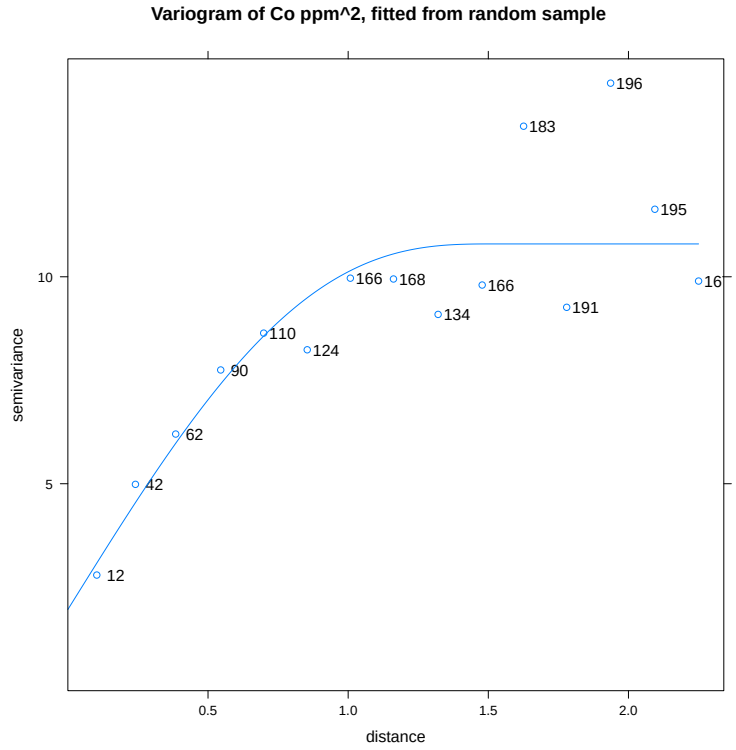
•

Task 7 : Model the variogram and compare this model with the known model. •

We could make the tentative model by eye, but the fit will be the same if we start from any "reasonable" model; we already have one of these, namely

the "true" model `vmf`:

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

  model  psill  range
1   Nug 1.9617 0.0000
2   Pen 8.8320 1.5203

> vmf

  model   psill  range
1   Nug  1.3712 0.0000
2   Pen 12.9322 1.5239

> print(plot(v, pl = T, model = vf, main = "Variogram of Co ppm^2, fitted from random sample
```



Q6 : *How close are the two models?* *Jump to A6* •

Q7 : *Why was it easy to model this variogram even with only 100 points?* *Jump to A7* •

2.4 Answers

A1 : *Spatial coverage is uneven; there are tight clusters, isolated points, large holes without any points nearby. For design-based sampling this is no problem; all*

sampling units are equally-likely. Return to Q1 •

A2 : *The answer here depends on the actual random sample. Typically a sample has a narrower range, since with only 100 points it is unlikely that we'll hit the extremes of 10 272 cells.*

The mean usually fairly close; but the inter-quartile range and variance may be wider or narrower. Return to Q2 •

A3 : *The two sample sizes are different: 100 vs. 10272, so they must be compared by their relative proportions.* Return to Q3 •

A4 : *The population histogram shows an almost perfect normal distribution; the sample is erratic. This is because of the sample size.* Return to Q4 •

A5 : *Again, this depends on the sample. In general, the model form (pentaspherical) seems adequate; there is no strong indication for another form (although perhaps spherical would fit well also). This experimental variogram is easy to model.* Return to Q5 •

A6 : *Quite close, especially considering the small number of sample points.* Return to Q6 •

A7 : *Because of the random sample, there are point-pairs at all distance ranges, so that all the experimental variogram bins have enough point-pairs for a good average.* Return to Q7 •

3 Grid sampling

Regular grids have the advantage of equal coverage. In a pure grid scheme, the only random element is the location of the first point; the others are at regular intervals.

The `spsample` method of the `sp` package can also draw a grid sample; to do this we specify the `type` argument as `"regular"`.

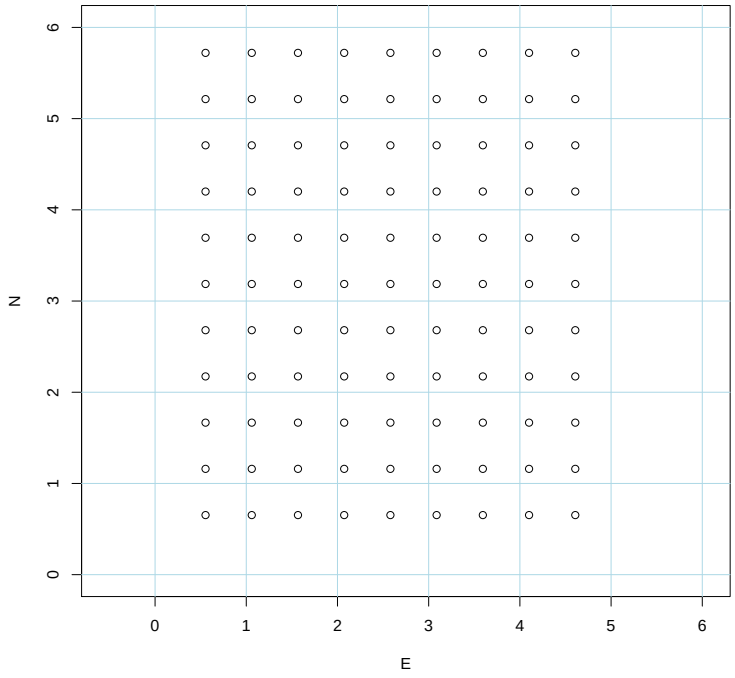
Task 8 : Place 100 sample points as a square grid over the prediction grid's bounding box; plot their locations. •

Unless the optional `offset` argument is specified, the first point of the square grid (say, the lower-leftmost) is placed randomly within the area it is to represent, and the rest of the grid is regular from that point. So that your grid looks the same as the one here, we specify the centre of each area, as `offset=c(0.5, 0.5)`.

> **Note:** You would specify a random offset with a call to the `runif` method, specifying the maximum dimensions of the offset, e.g. `offset=runif(2,`

`-.99, .99)` to draw two random offsets (for the two dimensions), each on $[-0.99 \ldots + 0.99]$, i.e. almost the corners of the cell.

```
> samp.grid <- spsample(jura.grid, n = 100, type = "regular",
+     offset = c(0.5, 0.5))
> str(samp.grid)

Formal class 'SpatialPoints' [package "sp"] with 3 slots
  ..@ coords     : num [1:99, 1:2] 0.553 1.06 1.567 2.074 2.58 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x1" "x2"
  ..@ bbox       : num [1:2, 1:2] 0.553 0.653 4.607 5.721
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x1" "x2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA

> plot(coordinates(samp.grid), asp = 1, xlab = "E", ylab = "N",
+     xlim = c(0, 5.5), ylim = c(0, 6))
> grid(lty = 1, col = "lightblue")
```



3.1 Grid sampling with jitter

A variant of this method is to add a small random "jitter" to each grid point. This ensures that we don't accidentally allign the grid with a systematic effect.

Task 9 : Add a normally-distributed "jitter" with standard deviation 20 m

to each grid point from the regular grid just computed; plot the sample locations. •

We extract the coördinates, add the jitter, and replace the grid coördinates with the perturbed ones. The manipulation of the coördinates must be done on columns of a data frame, so we first convert from a spatial object to a data frame with the `as.data.frame` method, then after the jitter is computed with the `rnorm` method and added to the original coordinates, convert back to a spatial object with the `coordinates` method.

```
> samp.grid <- as.data.frame(samp.grid)
> samp.grid$x1 <- samp.grid$x1 + rnorm(n = length(samp.grid$x1),
+     mean = 0, sd = 0.03)
> samp.grid$x2 <- samp.grid$x2 + rnorm(n = length(samp.grid$x2),
+     mean = 0, sd = 0.03)
```

3.2 Removing points outside the sample area

The jitter may have placed some points outside the simulation grid. We can identify these with the `which` method, which (!) returns the indices (positions in the vector) of the points which (!) meet a given logical criterion. We use the `|` ("or") operator to identify any coördinate outside the bounding box of the grid, which we extract with the `bbox` method:

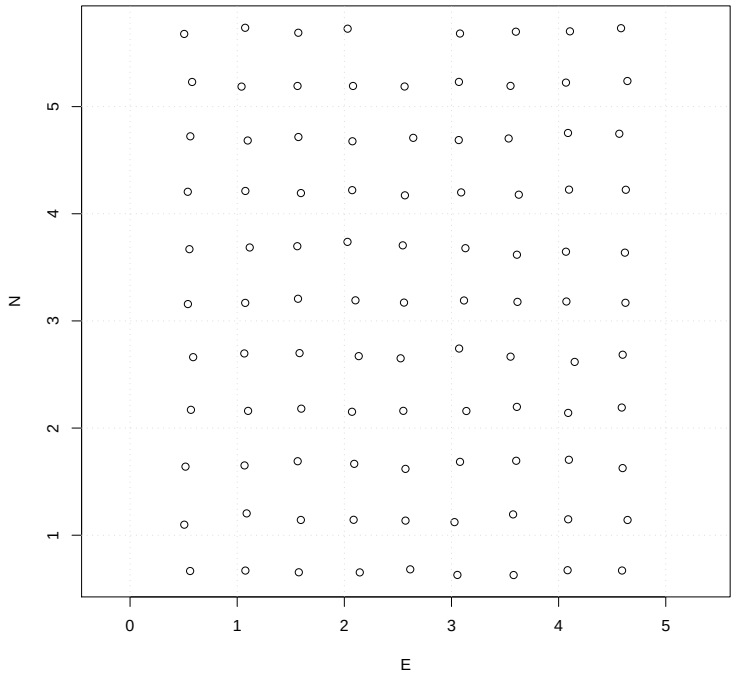
```
> (pts.outside <- which((samp.grid$x1 < bbox(k.sim)[1,
+     "min"]) | (samp.grid$x1 > bbox(k.sim)[1, "max"]) |
+     (samp.grid$x2 < bbox(k.sim)[2, "min"]) | (samp.grid$x2 >
+     bbox(k.sim)[2, "max"])))

[1] 95
```

Depending on the random noise with added to jitter the points, there may be no or several points in the list. If there are any such points we remove them, by selecting the rows in the set of row numbers, *without* the identified points; for this we use the `setdiff` "set difference" method.

```
> samp.grid <- samp.grid[setdiff(1:99, pts.outside), ]
> str(samp.grid)

'data.frame':        98 obs. of  2 variables:
 $ x1: num  0.56 1.08 1.58 2.14 2.61 ...
 $ x2: num  0.666 0.67 0.654 0.653 0.682 ...

> rm(pts.outside)
```

Now the grid can be converted to a spatial object and plotted:

```
> coordinates(samp.grid) <- ~x1 + x2

> plot(coordinates(samp.grid), asp = 1, xlab = "E", ylab = "N")
> grid()
```



Q8 : *Describe the spatial pattern of the perturbed grid sample.* *Jump to A8* •

3.3 * Overlaying with a bounding box

This **optional** section repeats the actions of the previous §3.3, i.e. removing jittered points outside the study area, but using a more 'spatial' method: polygon overlay. This GIS operation is included in the `sp` package. This section will be interesting to you if you intend to work with polygon coverages in `sp`, otherwise the simpler approach taken in the previous § should be sufficient.

There are three steps:

1. Make a spatial polygon object from the bounding box of the study area;
2. Identify the points outside this polygon.
3. Clip the point set to this polygon by removing those points.

Task 10 : Make a spatial polygon from the bounding box of the `jura.grid` study area. •

Spatial polygon objects in `sp` are polygon maps, i.e. they may contain many polygons, as in a GIS layer. They are of class `SpatialPolygons` and are

created with the `SpatialPolygons` method, which requires a **list** of polygon lists and the order in which they should be plotted.

In turn, each of these polygon lists is of class `Polygons` (note the "s"), which themselves are each a list of one or more polygons with a unique identifier. The primitive polygons are of class `Polygon`.

All this complexity is not needed for the simple case of a single polygon made from the bounding box! But the general structure allows complex polygon coverages to be created.

With that background, we build up the spatial polygon object step-by-step, to make the three levels clear.

First, a `Polygon` object, i.e. a single polygon, in this case of the bounding box. The polygon is defined by a matrix of coördinates (extracted with the `bbox` method), columnwise. So, we build up two lists, one for the East and one for the North, with two calls to the `c` method, and then bind the two vectors together columnwise with the `cbind` "column bind" method.

```
> poly.1 <- Polygon(cbind(c(bbox(jura.grid)[1, "min"],
+     bbox(jura.grid)[1, "min"], bbox(jura.grid)[1, "max"],
+     bbox(jura.grid)[1, "max"], bbox(jura.grid)[1, "min"]),
+     c(bbox(jura.grid)[2, "min"], bbox(jura.grid)[2, "max"],
+         bbox(jura.grid)[2, "max"], bbox(jura.grid)[2,
+             "min"], bbox(jura.grid)[2, "min"])))
> str(poly.1)

Formal class 'Polygon' [package "sp"] with 5 slots
  ..@ labpt  : num [1:2] 2.7 3.08
  ..@ area   : num 25.7
  ..@ hole   : logi FALSE
  ..@ ringDir: int 1
  ..@ coords : num [1:5, 1:2] 0.3 0.3 5.1 5.1 0.3 0.4 5.75 5.75 0.4 0.4
```

Second, a `Polygons` object, as a list of polygons with an identifier for the set; in this case just the one polygon:

```
> polys <- Polygons(list(poly.1), "boundary")
> str(polys)

Formal class 'Polygons' [package "sp"] with 5 slots
  ..@ Polygons :List of 1
  .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. ..@ labpt  : num [1:2] 2.7 3.08
  .. .. .. ..@ area   : num 25.7
  .. .. .. ..@ hole   : logi FALSE
  .. .. .. ..@ ringDir: int 1
  .. .. .. ..@ coords : num [1:5, 1:2] 0.3 0.3 5.1 5.1 0.3 0.4 5.75 5.75 0.4 0.4
  ..@ plotOrder: int 1
  ..@ labpt    : num [1:2] 2.7 3.08
  ..@ ID       : chr "boundary"
  ..@ area     : num 25.7
```

Third and finally, the `SpatialPolygons` object, as a list of `Polygons` objects with a plot order (here not relevant, so omitted):

```
> bboxPoly <- SpatialPolygons(list(polys))
> str(bboxPoly)

Formal class 'SpatialPolygons' [package "sp"] with 4 slots
  ..@ polygons   :List of 1
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 2.7 3.08
  .. .. .. .. .. .. ..@ area   : num 25.7
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:5, 1:2] 0.3 0.3 5.1 5.1 0.3 0.4 5.75 5.75 0.4 0.4
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 2.7 3.08
  .. .. .. ..@ ID       : chr "boundary"
  .. .. .. ..@ area     : num 25.7
  ..@ plotOrder  : int 1
  ..@ bbox       : num [1:2, 1:2] 0.3 0.4 5.1 5.75
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "r1" "r2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

This now has full topology. As you can see, the data structures get more complex as we build up from single polygons, to polygon lists, to a polygon layer.

The point of all this was to make a polygon that can clip the jittered points. Recall that we used the `overlay` method in §2.2 to overlay points and pixels; this generic method can also handle points and polygons.

Task 11 : Draw a completely random sample of 100 points from the `jura.grid` area, jitter them, and identify the points outside the study area. •

We first select the sample points as in §3.1 and then jitter them as in §, and then apply the `overlay` method:

```
> samp.grid <- spsample(jura.grid, n = 100, type = "regular",
+     offset = c(0.5, 0.5))
> samp.grid <- as.data.frame(samp.grid)
> samp.grid$x1 <- samp.grid$x1 + rnorm(n = length(samp.grid$x1),
+     mean = 0, sd = 0.05)
> samp.grid$x2 <- samp.grid$x2 + rnorm(n = length(samp.grid$x2),
+     mean = 0, sd = 0.05)
> coordinates(samp.grid) <- ~x1 + x2
```

When applied to points and polygons, `overlay` returns a vector of the same length as the point set, containing the polygon number in which it falls, or `NA` if none. In this case there's only one polygon (the bounding box), and all points identified with `NA` are outside of it. We count these with the `sum` method applied to the logical vector created by the `is.na` method:

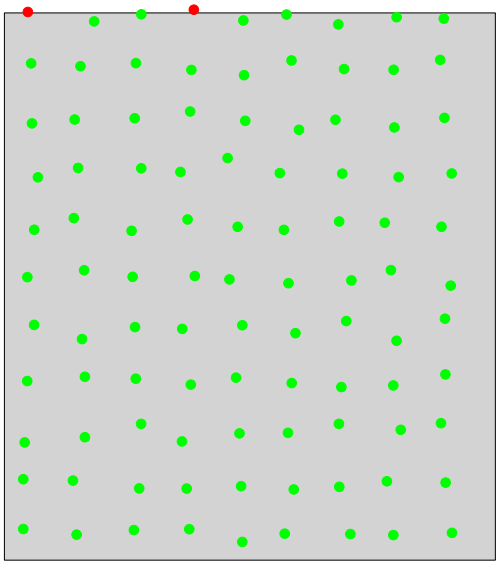
```
> ov <- overlay(samp.grid, bboxPoly)
> sum(is.na(ov))
```

```
[1] 2

> samp.grid[is.na(ov)]

SpatialPoints:
          x1     x2
[1,] 0.52937 5.7593
[2,] 2.15227 5.7816
Coordinate Reference System (CRS) arguments: NA

> bbox(jura.grid)

   min  max
s1 0.3 5.10
s2 0.4 5.75
```

Q9 : *How many jittered points are outside the study area? How can you confirm this from the bounding box?* *Jump to A9* •

We can see this visually by plotting the jittered sample points, their bounding box, and the bounding box of the study area, with rejected points coloured red:

```
> print(plot(bboxPoly, col = "lightgray"))

NULL

> points(samp.grid, col = ifelse(is.na(ov), "red", "green"),
+     pch = 20, cex = 2)
```



Task 12 : Remove the points outside the study area from the sample set. •

We again use the `is.na` method to find the index numbers of the rejected points, but then invert the logical sense with the `!` ("not") logical operator:

```
> samp.grid <- samp.grid[!is.na(ov)]
> summary(samp.grid)

Object of class SpatialPoints
Coordinates:
       min    max
x1 0.48455 4.6755
x2 0.57850 5.7366
Is projected: NA
proj4string : [NA]
Number of points: 97
```

Task 13 : Clean up from this section, leaving the (jittered) sample grid object. •

```
> rm(poly.1, polys, bboxPoly, ov)
```

3.4 Evaluation: reproducing population statistics

We repeat the evaluation of the sample, first with the population statistics:

Task 14 : Extract the simulated Co values at the sample points. •

```
> ov <- overlay(k.sim, samp.grid)
> str(ov)

 int [1:97] 9604 9807 9722 9637 9935 9751 9764 9869 9784 8740 ...

> Co.grid <- k.sim$sim1[ov]
> str(Co.grid)

 num [1:97] 4.74 7.89 9.08 4.42 5 ...
```

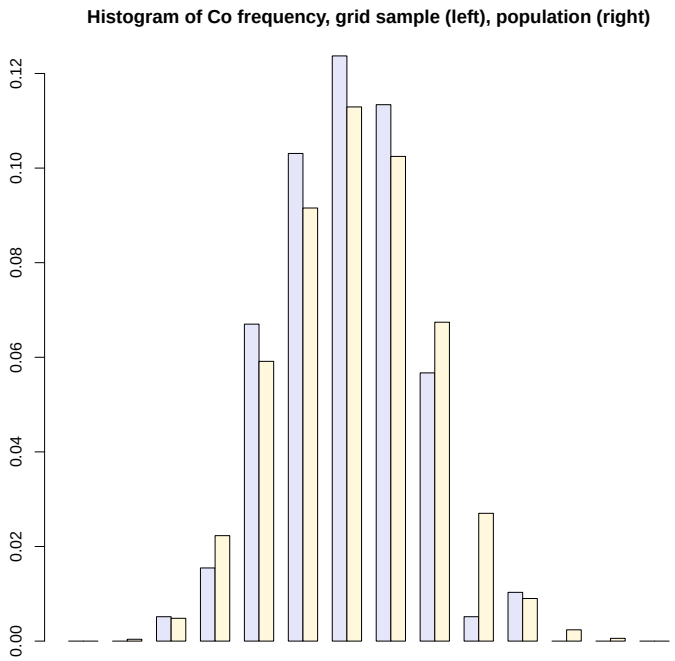
Q10 : *How well does this sample reproduce the population mean, variance, extremes, and quartiles?* *Jump to A10* •

```
> summary(Co.grid)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.19    7.12    8.96    8.93   11.10   17.30

> summary(k.sim$sim1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -1.76    6.91    9.28    9.28   11.60   21.40

> var(Co.grid)

[1] 9.0133
```

```
> var(k.sim$sim1)

[1] 11.225
```

Task 15 : Display the distribution of the sample, alongside that of the full distribution, as a side-by-side frequency histogram. •

```
> hist.grid <- hist(Co.grid, breaks = seq(-4, 24, by = 2),
+     plot = F)
> hist.sim <- hist(k.sim$sim1, breaks = seq(-4, 24, by = 2),
+     plot = F)
> to.plot <- rbind(samp = hist.grid$density, pop = hist.sim$density)
> barplot(to.plot, beside = T, col = c("lavender", "cornsilk"),
+     main = "Histogram of Co frequency, grid sample (left), population (right)")
> rm(hist.grid, hist.sim, to.plot)
```

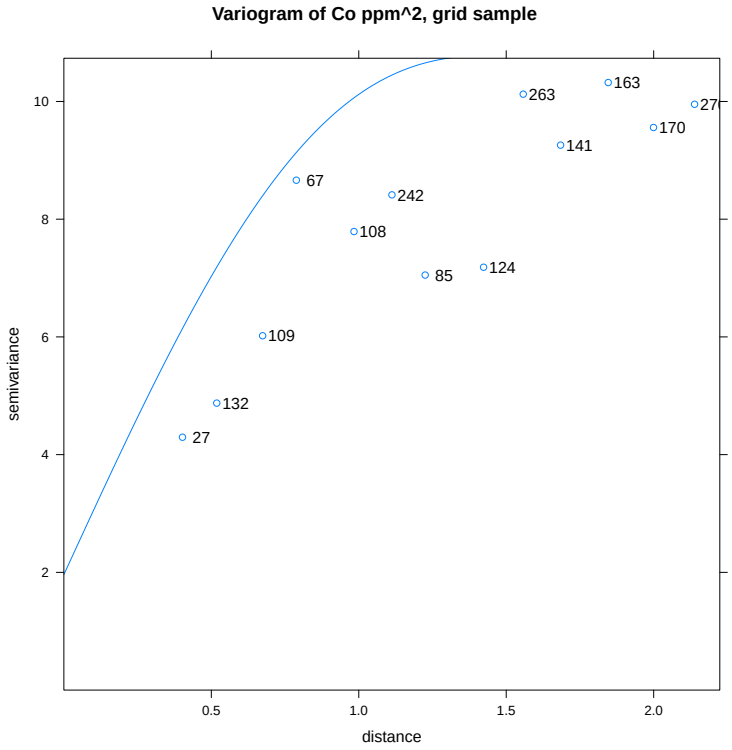


3.5 Evaluation: reproducing the variogram

We repeat the evaluation of the sample, second with the experimental variogram:

Task 16 : Compute the experimental variogram for Co at the sample points. Plot it with the known model. •

```
> Co.grid <- data.frame(Co = Co.grid)
> coordinates(Co.grid) <- coordinates(samp.grid)

> v <- variogram(Co ~ 1, loc = Co.grid)
> print(plot(v, pl = T, model = vf, main = "Variogram of Co ppm^2, grid sample"))
```



Q11 : *How well is the experimental variogram computed from the sample fitted by the known model? Would another model form appear more appropriate? Can this experimental variogram be modelled?* *Jump to A11*
•

3.6 Answers

A8 : *The points cover the area fairly evenly; there are no large holes and no clusters.* *Return to Q8* •

A9 : *This depends on your random sample (you can try it several times to compare); with this much jitter about 1% to 20% of the points will be outside. Confirm this by comparing to the corner values of the bounding box.* *Return to Q9* •

A10 : *Quite similar results to the 100 random points.* *Return to Q10* •

A11 : *The model fits the variogram fairly well, but there are no variogram points closer than 0.5 km. Thus the estimate of a model form and nugget are wild speculation. It is not appropriate to use this experimental variogram as the basis of a model.* *Return to Q11* •

4 Optimal grid sampling

We saw in the previous section that a grid sample is not a good way to estimate a variogram. However, if the variogram model is **known** from previous studies, or **assumed** from literature or even an educated guess, a grid sample is provably the most efficient for **mapping** an area, as shown by McBratney and Webster [8, 7]. Their method is known as OSSFIM: "optimal sampling scheme for isarithmic mapping" and is implemented by the `ossfim` method of `gstat`.

> **Note:** There is a simple web-based interactive version of OSSFIM at http://www.gstat.org/ossfim.html.

For any "optimal" method we need to define an objective function; here it is to minimize the maximum kriging variance of any prediction point. These are of course the points at the centres of the square (or triangular) tesselation.

Three aspects of kriging affect the kriging variance at a point:

1. Sample configuration; here we have decided for a square grid of **unknown side**;
2. Model of spatial dependence; here we have a known variogram model;
3. Block size; the larger the block average, the lower the kriging variance.

OSSFIM allows us to trade off block size and sample spacing.

Task 17 : Compute the kriging variance for sample spacings of 100 m, 150 m, ... 400 m, and for block sizes of (0 m)^2(i.e. punctual kriging), (20 m)^2, ... (100 m)^2. •

To match the model we express the distances in kilometers:

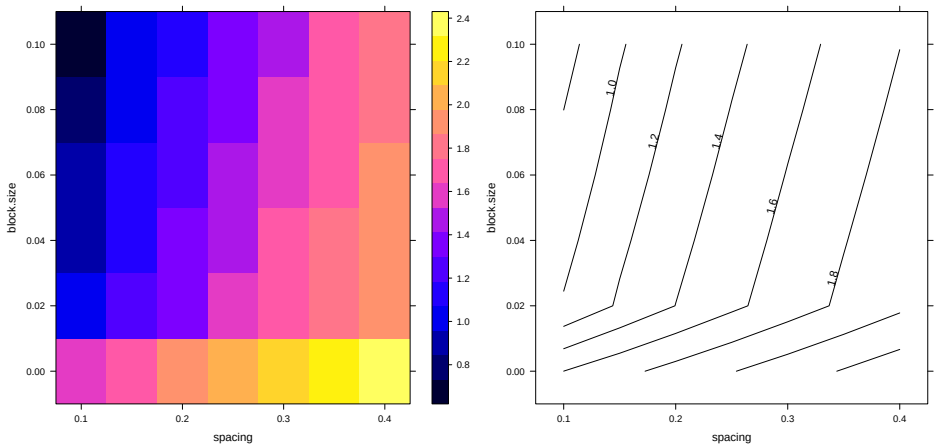
```
> err.grid <- ossfim(seq(0.1, 0.4, 0.05), seq(0, 0.1, 0.02),
+     vmf)
> str(err.grid)

'data.frame':        42 obs. of  3 variables:
 $ spacing   : num  0.1 0.1 0.1 0.1 0.1 0.1 0.15 0.15 0.15 0.15 ...
 $ block.size: num  0 0.02 0.04 0.06 0.08 0.1 0 0.02 0.04 0.06 ...
 $ kriging.se: num  1.601 1.016 0.943 0.871 0.8 ...
```

Task 18 : Plot the kriging variance as a function of spacing and block size. •

Both the `levelplot` and `contourplot` methods give useful views:

```
> plot.1 <- levelplot(kriging.se ~ spacing + block.size,
+     err.grid, col.regions = bpy.colors(64))
> plot.2 <- contourplot(kriging.se ~ spacing + block.size,
+     err.grid)
> print(plot.1, split = c(1, 1, 2, 1), more = T)
```

```
> print(plot.2, split = c(2, 1, 2, 1), more = F)
> rm(plot.1, plot.2)
```



Q12 : *What happens to the kriging variance at the central point of a grid cell as the sample spacing increases? As the block size increases? Which of these effects is greater, with the spacings and sizes we've chosen?* *Jump to A12* •

We can see this numerically as well as graphically. First, the effect of increasing spacing:

```
> err.grid[err.grid$block.size == 0, ]
```

```
   spacing block.size kriging.se
1     0.10          0     1.6006
7     0.15          0     1.7417
13    0.20          0     1.8704
19    0.25          0     1.9905
25    0.30          0     2.1044
31    0.35          0     2.2135
37    0.40          0     2.3192
```

Second, the effect of increasing block size:

```
> err.grid[err.grid$spacing == 0.1, ]
```

```
  spacing block.size kriging.se
1     0.1       0.00    1.60061
2     0.1       0.02    1.01647
3     0.1       0.04    0.94285
4     0.1       0.06    0.87051
5     0.1       0.08    0.79960
6     0.1       0.10    0.73055
```

Suppose we have a target maximum kriging prediction standard error (perhaps from a project specification). Then we can see the trade-off between spacing and block size that allows us to reach that target.

Task 19 : Find all combinations of spacings and block sizes that reduce

the kriging prediction standard error below 1.5 mg kg^{-1}, and sort them in decreasing order of standard error. •

```
> tmp <- err.grid[err.grid$kriging.se < 1.5, ]
> tmp[order(tmp$kriging.se), ]
```

	spacing	block.size	kriging.se
6	0.10	0.10	0.73055
5	0.10	0.08	0.79960
4	0.10	0.06	0.87051
3	0.10	0.04	0.94285
12	0.15	0.10	0.97758
2	0.10	0.02	1.01647
11	0.15	0.08	1.03874
10	0.15	0.06	1.10054
9	0.15	0.04	1.16295
18	0.20	0.10	1.18034
8	0.15	0.02	1.22591
17	0.20	0.08	1.23535
16	0.20	0.06	1.29069
15	0.20	0.04	1.34634
24	0.25	0.10	1.35632
14	0.20	0.02	1.40227
23	0.25	0.08	1.40665
22	0.25	0.06	1.45716

```
> rm(tmp)
```

Q13 : *What is the widest spacing we can use and still achieve the target? What is the minimum block size we can use at this spacing?* *Jump to A13*
•

Task 20 : Clean up from this section. •

```
> rm(samp.rand, samp.grid, Co.rand, Co.grid, ov, v, vf,
+     err.grid)
```

4.1 Answers

A12 : *Increasing spacing: increases; increasing block size: decreases; the second effect is more dramatic; i.e. if block averages are sufficient information the spacing can be much wider with the same uncertainty.* *Return to Q12* •

A13 : *The widest spacing is 0.25 km, i.e. 250 m; at this spacing the smallest block is (0.06 km)^2, i.e. (60 m)^2.* *Return to Q13* •

5 Self-test

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

(including graphical output) to the instructor for grading and sample answers.

Task 1 : Draw **two different samples** of size 50 **either** of a completely random **or** a grid design from the simulated field of §1, and compare the summary statistics, histograms and variograms for Co concentration.

(If you are ambitious you can compare both the completely random and grid samples.) •

Q1 : *How different are the two samples from each other? Comment on the statistics, histograms, and how well they reproduce the known variogram.* •

Task 2 : Repeat for size 500. •

Q2 : *How different are the two samples from each other? Comment on the statistics, histograms, and how well they reproduce the known variogrqam.* •

Q3 : *Are the 500-unit samples more or less consistent than the 50-unit samples? Why or why not?* •

References

[1] D J Brus and J J de Gruijter. Random sampling or geostatistical modelling? Choosing between design-based and model-based sampling strategies for soil (with Discussion). *Geoderma*, 80(1-2):1–59, 1997. 3

[2] D J Brus and G B M Heuvelink. Optimization of sample patterns for universal kriging of environmental variables. *Geoderma*, 138(1-2):86–95, 2007. 1

[3] W G Cochran. *Sampling Techniques*. John Wiley, New York, 3rd edition, 1977. 3

[4] J de Gruijter, D J Brus, M F P Bierkens, and M Knotters. *Sampling for Natural Resource Monitoring*. Springer, 2006. 1

[5] T Hengl, D G Rossiter, and A Stein. Soil sampling strategies for spatial prediction by correlation with auxiliary maps. *Australian Journal of Soil Research*, 41(8):1403 – 1422, 2003. 1

[6] R M Lark. Optimized spatial sampling of soil for estimation of the variogram by maximum likelihood. *Geoderma*, 105(1-2):49–80, 2002. 1

[7] A B McBratney and R Webster. The design of optimal sampling schemes for local estimation and mapping of regionalized variables - II. *Computers & Geosciences*, 7(4):335–365, 1981. 20

[8] A B McBratney, R Webster, and T M Burgess. The design of optimal sampling schemes for local estimation and mapping of regionalized variables - I. *Computers & Geosciences*, 7(4):331–334, 1981. 20

[9] B Minasny, A B. McBratney, and D J J Walvoort. The variance quadtree algorithm: Use for spatial sampling design. *Computers & Geosciences*, 33(3):383–392, 2007. 1

[10] W G Müller and D L Zimmerman. Optimal designs for variogram estimation. *Environmetrics*, 10(1):23–37, 1999. 1

[11] D Russo. Design of an optimal sampling network for estimating the variogram. *Soil Science Society of America Journal*, 48(4):708–716, 1984. 1

[12] H T Schreuder, R Ernst, and H Ramirez-Maldonado. *Statistical techniques for sampling and monitoring natural resources*. Gen. Tech. Rep. RMRS-GTR-126. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, Fort Collins, CO, 2004. URL `http://www.fs.fed.us/rm`. 1

[13] A Stein and C Ettema. An overview of spatial sampling procedures and experimental design of spatial studies for ecosystem comparisons. *Agriculture, Ecosystems & Environment*, 94(1):31–47, 2003. 1

[14] J W van Groenigen, A Stein, and R Zuurbier. Optimization of environmental sampling using interactive gis. *Soil Technology*, 10(2):83–97, 1997. 1

[15] A W Warrick and D E Myers. Optimization of sampling locations for variogram calculations. *Water Resources Research*, 23(3):496–500, 1987. 1

[16] R. Webster, S. J. Welham, J. M. Potts, and M. A. Oliver. Estimating the spatial scales of regionalized variables by nested sampling, hierarchical analysis of variance and residual maximum likelihood. *Computers & Geosciences*, 32(9):1320–1333, 2006. 1

Index of R Concepts

as.data.frame, 11

barplot, 5
bbox (package:sp), 12, 14

c, 14
contourplot (package:lattice), 20
coordinates (package:sp), 3, 4, 7, 11

data.frame, 7

gstat package, 20

hist, 5

is.na, 15, 17

levelplot (package:lattice), 20

ossfim (package:gstat), 20
overlay (package:sp), 5, 15

Polygon (sp class), 14
Polygons (sp class), 14

rnorm, 11
runif, 10

set.seed, 2
setdiff, 12
sp package, 3, 5, 10, 13
SpatialPixelsDataFrame (sp class), 5
SpatialPoints (sp class), 3, 5
SpatialPolygons (package:sp), 14
SpatialPolygons (sp class), 13, 14
spplot (package:sp), 2
spsample (package:sp), 3, 10
sum, 15

which, 12