Applied geostatistics
Exercise 10: Change of support

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

January 6, 2014

Contents

苟不教性乃迁(遷)
"If, due to carelessness, a person is not taught, his nature will deteriorate."
– 三字经 ("Three-character classic", 1)

1 Introduction

All geographical measurements are made on some **support**, that is, an interval (1-D), area (2-D) or volume (3-D) of some finite size. As long as the measurements, interpretations, and predictions all refer to the same support, techniques that treat the support as a 0-D point are satisfactory. But if measurements and predictions are made on different supports, the relation between them must be determined and used to adjust the geostatistical analysis.

The issue of geostatistical support is discussed in many texts, e.g. Isaaks and Srivastava [5, §19] and Bivand et al. [1, §5.1], especially those aimed at mining geostatistics, e.g. Rendu [9, §5]. The treatment here uses the notation of Webster and Oliver [14, §4.8].

After completing this exercise you should be able to:

1. Determine the dispersion variance at different supports;
2. Adjust variograms from one support to another.

Note: This text is not complete (hence the version number < 1); in particular, no answers to questions have been written.

2 The regularized variable

The concept underlying change of support is the so-called **regularized** spatial variable [3]. This is the average value of a variable $\mathbf{x}$ over some physical volume B. This is easily defined as:

$$\mathbf{x}(B) = \frac{1}{|B|}\int_B \mathbf{x}(s)ds \tag{1}$$

where s is conceptually a 0-dimensional point. However, the actual integration is not straightforward, because:

1. The conceptually infinite block B must be **discretized**;
2. There is generally **spatial structure** within the block: points are not independent, but rather spatially auto-correlated, as revealed by variogram or correlogram analysis;
3. The actual supports of the measurements to be integrated may be too large to be considered approximations to 0-dimensional points, relative to the scale of integration (the volume B).

A special case of change of support, avoiding problem (3) of the above list, is **block kriging**, covered in Exercise 5 of this series. This is when the sample

support is small enough, relative to the block size, to be considered as punctual (a 0-dimensional point), but the prediction is needed for larger supports, called **blocks**. In this case the variogram for a punctual support is used to integrate a spatially-averaged prediction for each block. This is a common technique for **upscaling** (e.g., [10, 11]).

However, this is not satisfactory if the sample support is an appreciable fraction of the target block size, so that the sample can not be considered punctual. It is also not applicable for the reverse case, i.e. **downscaling**, where geostatistical predictions are needed for smaller areas than the sample support.

As will be shown below (§5), changing the support of a variable through regularization creates a new and different variable, which thus has different non-spatial and spatial statistics.

> **Note:** In geostatistics, the blocks generally form a regular tesselation of the study area. In geography, arbitrary polygons may be formed to group observations. This is called the *Modifiable Areal Unit Problem* [1, 7] and is discussesed by Gotway Crawford and Young [3] as part of the concept of support in a wider context than geostatistics.

3 Dispersion variance

As preparation for understanding how support affects (geo-)statistical analysis, we first must understand how the variance of a random function changes with support within a finite region. The technical term used is **dispersion variance**; is defined as ([14, §4.8]):

Dispersion variance

$$\sigma_R^2 = \overline{\gamma}(R,R) = \frac{1}{|R|^2}\int_R\int_R \gamma(\mathbf{x}-\mathbf{x}')d\mathbf{x}d\mathbf{x}' \tag{2}$$

where R represents the entire region. This formula expresses that every point in the region is compared to every other point, and their semivariances are averaged. Of course, in practice the region is **discretized** into some number of points, and the double integral is replaced by a double summation.

Q1 : *Under what circumstances is this the same as average variance of all the discretization points? Why is it not always the same?* *Jump to A1* •

This σ_R^2 is the **upper limit** of variance across a region. The **lower limit** is the dispersion variance of the **sampling support**, i.e. the size of the physical entity that is actually sampled. These discretize the region R into n_R^b cells; for example, counts of a given plant species in a 1x1 m square (the support) discretize a hectare study area into 10 000 cells. Given a study area and support size, we can count the possible supports that cover the area and determine their variance within R:

Dispersion variance of a region

$$s^2(b \in R) = \frac{1}{n_R^b}\sum_{i=1}^{n_R^b}[\overline{z}_R - z_i^b]^2 \tag{3}$$

where $\overline{z}_R$ is the mean of the z_i^b, i.e. of the value of the target variable averaged over cell i. The latter is the value that is obtained by sampling and measuring at the sampling support. This dispersion variance is necessarily smaller than the theoretical dispersion variance of the region, because the portion of the variance within the cells has been removed by sampling and measuring at points that are, by necessity, of some physical size (finite support), not 0-dimensional points.

We may be interested in **blocks** larger than the sampling support, e.g., for prediction of averages in blocks of this size. Each of these has the same (finite) number of cells, n_B^b. First, we consider the variance of the cells within one of these blocks B. This is in principle computable, by exhaustively sampling a block. By analogy to Equation 3:

Dispersion variance within a block

$$s^2(b \in B) = \frac{1}{n_B^b} \sum_{j=1}^{n_B^b} [\overline{z}_B - z_j^b]^2 \tag{4}$$

where $\overline{z}_B$ is the within-block average of the n_B^b cells within the block, and z_j^b is the value of the target variable in cell j. If we do this for all the k blocks in the region, their within-block variances, as computed by Equation 4, can be averaged:

Average dispersion variance within blocks

$$\overline{s}^2(b \in B) = \sum_{k=1}^{n_R^B} s^2(b \in B_k) \tag{5}$$

This is the average dispersion variance of a block of support B, discretized into cells of sampling supports b, and is an estimate of the dispersion variance between these two supports.

Finally, consider the blocks within the region. Their within-block averages also have a variance, again by analogy to Equation 3:

Dispersion variance of blocks in a region

$$s^2(B \in R) = \frac{1}{n_R^B} \sum_{k=1}^{n_R^B} [\overline{z}_R - \overline{z}_k^B]^2 \tag{6}$$

where here $\overline{z}_R$ is the mean of the $\overline{z}_k^B$, which are the values of the target variable averaged over one of the n_R^B blocks in the region.

All of this prepares us for the partitioning of variance by **Krige's relation**:

$$s^2(b \in R) = s^2(B \in R) + \overline{s}^2(b \in B) \tag{7}$$

which just says that the overall dispersion variance of the block size of interest is the sum of two nested components: the blocks within the region, and the average, over all blocks, of the cells within these blocks. This relation can be used to understand scales of variation, and to convert between supports, as shown later.

How can these relations be computed, short of exhaustive sampling? The key is the model of spatial dependence. The expected values of these variances

are all directly related to the variogram:

$$\sigma^2(b \in R) = \overline{\gamma}(R,R) - \overline{\gamma}(b,b) \quad (8)$$

$$\sigma^2(B \in R) = \overline{\gamma}(R,R) - \overline{\gamma}(B,B) \quad (9)$$

$$\sigma^2(b \in B) = \overline{\gamma}(B,B) - \overline{\gamma}(b,b) \quad (10)$$

Note that in these equations the theoretical variance σ^2, based on the assumed random field (as expressed in the variogram model), replaces the empirical variance s^2. So if we have a variogram model of the random field that is assumed to model the variable's spatial dependence, we can compute the required semivariances by integrating over the required support. In practice, this is done by discretizing the support into a "large" number of points, and computing the average semivariance between all of these, according to the variogram model.

For example, the within-block semivariance $\overline{\gamma}(b,b)$ is estimated by analogy to Equation 14 as:

$$\overline{\gamma}(b,b) = \frac{1}{|b|^2} \int_b \int_b \gamma(\mathbf{x} - \mathbf{x}') d\mathbf{x} d\mathbf{x}' \quad (11)$$

This could be discretized by computing punctual semivariances between all pairs of n points in the block:

$$\overline{\gamma}(b,b) \approx \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma(\mathbf{x_i}, \mathbf{x_j}) \quad (12)$$

The semivariance is of course a function of the separation (and perhaps azimuth) between discretizing points. In practice, a fairly small number of points (9, 16, or 25) arranged on a regular grid (rectangular or hexaogonal) sufficies to reliabily approximate the average.

4 Computing dispersion variances

To see how the above relations work, we compute them on a known random field. This has the advantage that we know exactly what the relations should be, from the known model of spatial dependence.

4.1 Simulating a random field

The known random field is created by **stochastic simulation** from a known model of spatial dependence [1, §8.7].

The first step is to create a regular grid, over which the random field will be simulated.

Task 1 : Create a 128x128 grid of unit cells. •

The `GridTopology` method is used to define the coordinates of a grid, and the `SpatialGrid` method to create an object of class `SpatialGrid` with grid topology.

```
> grid.128 <- SpatialGrid(GridTopology(cellcentre.offset = c(0.5,
+     0.5), cellsize = c(1, 1), cells.dim = c(128, 128)))
> str(grid.128)

Formal class 'SpatialGrid' [package "sp"] with 3 slots
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  .. .. ..@ cellcentre.offset: num [1:2] 0.5 0.5
  .. .. ..@ cellsize         : num [1:2] 1 1
  .. .. ..@ cells.dim        : int [1:2] 128 128
  ..@ bbox       : num [1:2, 1:2] 0 0 128 128
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

Second, specify a variogram model.

Task 2 : Select a variogram model to represent the spatially-correlated random function for simulation. •

We choose a simple exponential model with unit total sill (the units are arbitrary) and range parameter 8, i.e. an effective range of $3 \times 8 = 24$, which is about $1/5$ of each dimension of the grid. This shows good spatial dependence while still allowing for variation across the grid. We also include a nugget effect of $1/5$ of the total sill, to simulate a variable with some variability within the support; later we'll see the effect of increasing support.

```
> vm <- vgm(psill = 0.8, , model = "Exp", range = 8, nugget = 0.2)
```

Third, simulate a realization of the random field.

Task 3 : Simulate a random field with the selected variogram model. •

We use the `krige` comman with parameter `loc=NULL` to specify an **unconditional** simulated field, with known structure given by a variogram model, specified with `vgm`. The `set.seed` command specifies a starting point for the random number generator so that your results match those shown here. In addition, the optional `maxdist` "maximum distance" argument limits the simulation to this radius; otherwise even for this 128x128 = 16 384 cell field the simulation is extremely slow. We select `maxdist` to cover the effective range of the variogram, i.e. 24 cells.

> **Note:** This simulation may take some time; we check it with the `system.time` function.

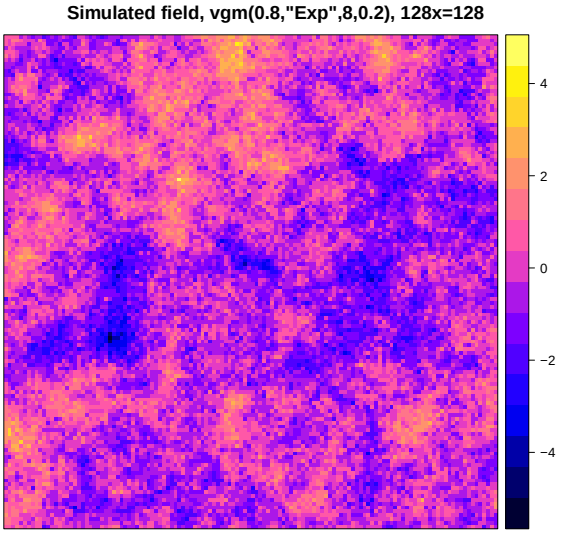
```
> set.seed(621)
> system.time(
+             k.e8.128 <- krige(z~1, loc=NULL,
+                               newdata=grid.128,
+                               model=vm, beta=0,
+                               dummy=T, nsim=1,
```

```
+                                maxdist=24)
+             )
> names(k.e8.128) <- "z"

[using unconditional Gaussian simulation]
   user  system elapsed
7356.148    66.185 13731.566
```

On my system this took 3 hr 49 min. If your system is slow or if you are impatient, you can reduce `maxdist` but will not reproduce the known variogram at longer ranges (§5.2).

Task 4 : Summarize and visualize the simulated random field. •

```
> summary(k.e8.128)

Object of class SpatialGridDataFrame
Coordinates:
   min max
s1   0 128
s2   0 128
Is projected: NA
proj4string : [NA]
Grid attributes:
   cellcentre.offset cellsize cells.dim
s1               0.5        1       128
s2               0.5        1       128
Data attributes:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -5.000  -0.914  -0.239  -0.230   0.441   4.400

> sim.plot.128 <- spplot(k.e8.128, col.regions = bpy.colors(64),
+     main = "Simulated field, vgm(0.8,\"Exp\",8,0.2), 128x=128")
> print(sim.plot.128)
```



4.2 Dispersion variances in the random field

Before considering dispersion variance, we first compute the naïve variance, i.e. not taking into account spatial structure, to compare with the dispersion variances of various-size subregions.

Q2 : *What is the naïve variance of this simulated field?* *Jump to A2* •

```
> var(k.e8.128$z)

[1] 1.0404
```

We consider the sampling support (1 x 1) to be effectively punctual, and so do not consider the dispersion variance within these small supports. In practice these are the the sampling units.

> **Note:** If we analyze the spatial dependence of a set of observations on this support, and use this to predict at other points with the same support (for example, by punctual Ordinary Kriging, OK), or block averages at a larger support (by Block Ordinary Kriging, BOK), there is no issue with support: everything is on the same support. Although the prediction blocks are larger than the point support, BOK predicts at "all" points (in practice, some discretization) in the block by punctual OK, and then averages these. The prediction variance is, however, lowered by the dispersion variance within the block, $\overline{\gamma}(b, b)$ from Equation 8.

For some computations it is convenient to also have this object as class `SpatialPointsDataFrame`, i.e. separate points instead of a grid.

Task 5 : Convert the simulated field to a spatial points data frame. •

```
> k.e8.128.pts <- SpatialPointsDataFrame(SpatialPoints(k.e8.128),
+     data = k.e8.128@data)
> str(k.e8.128.pts)

Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':        16384 obs. of  1 variable:
  .. ..$ z: num [1:16384] 0.974 0.987 0.556 0.539 -0.697 ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:16384, 1:2] 0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "s1" "s2"
  ..@ bbox       : num [1:2, 1:2] 0.5 0.5 127.5 127.5
  .. ..- 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
```

4.2.1 Dispersion variance of 2 x 2 blocks

We start with the smallest block larger than the sampling support.

Task 6 : Compute and summarize the dispersion variance at the smallest block support, i.e. 2 x 2. •

The `variogram` method, with the `cloud=T` optional argument, returns the semivariance of *all* point-pairs within the cutoff, specified with the optional `cutoff` argument.

We extract windows from the whole area with the `subset` command, specifying the coordinates to be selected with the `%in%` set operator, and combining the criteria for the two coordinate dimensions with the `&` logical operator.

For example, the upper-left 2 x 2 block is:

```
> (block <- subset(k.e8.128.pts, (coordinates(k.e8.128.pts)[,
+     1] %in% c(0.5, 1.5)) & (coordinates(k.e8.128.pts)[,
+     2] %in% c(0.5, 1.5))))

      coordinates       z
16129  (0.5, 1.5) 1.45655
16130  (1.5, 1.5) 1.40355
16257  (0.5, 0.5) 0.98993
16258  (1.5, 0.5) 0.72709
```

Its variogram cloud is:

```
> (vc <- variogram(z ~ 1, loc = block, cutoff = sqrt(2),
+     cloud = T))

    dist     gamma dir.hor dir.ver   id left right
1 1.0000 0.0014045       0       0 var1    2     1
2 1.0000 0.1088663       0       0 var1    3     1
3 1.4142 0.0855397       0       0 var1    3     2
4 1.4142 0.2660533       0       0 var1    4     1
5 1.0000 0.2287961       0       0 var1    4     2
6 1.0000 0.0345419       0       0 var1    4     3
```

> **Note:** The optional `cutoff` argument was necessary because by default `variogram` sets the cutoff to 1/3 of the distance across the bounding box; instead we set it to the diagonal across the box, $\sqrt{2}$.

Q3 : *How many point-pairs are there in this 2 x 2 block? What are the separations, and how many point-pairs have each separation?* *Jump to A3*
•

The dispersion variance can be directly computed from this as the average of the individual semivariances:

```
> mean(vc$gamma)

[1] 0.12087
```

The same figure can be arrived at with the empirical averaged semivariogram, as the weighted average of the bins, taking advantage of R's vectorized operators:

```
> (v <- variogram(z ~ 1, loc = block, cutoff = 2 * sqrt(2)))

  np   dist    gamma dir.hor dir.ver   id
1  4 1.0000 0.093402       0       0 var1
2  2 1.4142 0.175796       0       0 var1

> sum(v$np * v$gamma)/sum(v$np)

[1] 0.12087
```

The dispersion variances for this support are in general different for each 2 x 2 block. We can compute the variances for all possible 2 x 2 blocks and summarize, using the `for` looping operator. It's not necessary to do this for the full area, since a representative sub-area covering the variogram range will give almost identical summary statistics. We choose the upper-left corner, to the effective variogram range (24 units); this contains $12 \times 12 = 144$ non-overlapping 2 x 2 blocks, each of which dispersion variance must be calculated from its empirical variogram.

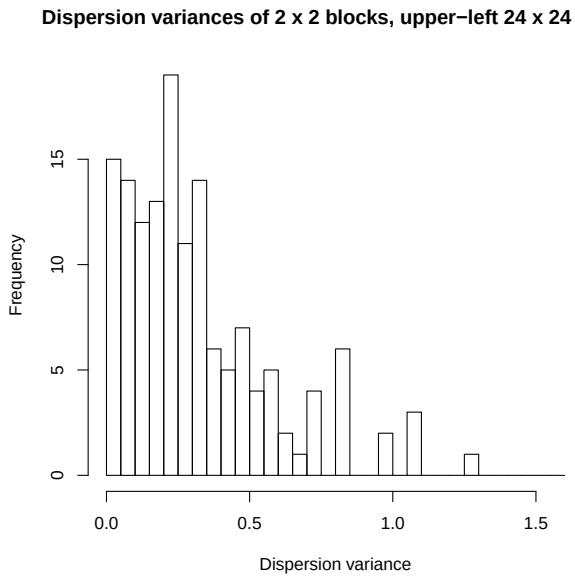
```
> bbox(k.e8.128.pts)

   min   max
s1 0.5 127.5
s2 0.5 127.5

> dv.2 <- vector(mode = "numeric", length = 12^2)
> i <- 1
> for (x in seq(0.5, 23.5, by = 2)) {
+     for (y in seq(0.5, 23.5, by = 2)) {
+         block <- subset(k.e8.128.pts, (coordinates(k.e8.128.pts)[,
+             1] %in% c(x, x + 1)) & (coordinates(k.e8.128.pts)[,
+             2] %in% c(y, y + 1)))
+         v <- variogram(z ~ 1, loc = block, cutoff = 2 *
+             sqrt(2))
+         dv.2[i] <- sum(v$np * v$gamma)/sum(v$np)
+         i <- i + 1
+     }
+ }
> length(dv.2)

[1] 144

> head(dv.2)

[1] 0.12087 0.32007 0.21180 0.74266 0.25091 0.34509
```

> **Note:** The `vector` function was used to initialize the results vector; the computation is faster when the result of one loop is placed in an existing vector slot, rather than when a vector is extended with the `c` function.

We summarize the dispersion variances on this support, and display as a histogram:

```
> summary(dv.2)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00325 0.13200 0.24500 0.32100 0.43600 1.29000
```

```
> hist(
+     dv.2,
+     main="Dispersion variances of 2 x 2 blocks, upper-left 24 x 24 block",
+     xlab="Dispersion variance", breaks=seq(0,1.6,by=.05))
```



The average dispersion variance at this support is shown in the summary as 0.3206.

Q4 : *What is the distribution of the dispersion variances of the 2 x 2 blocks?*
Jump to A4 •

Clearly the distribution is strongly right-skewed, with a few unfortunate high variances and many low, which is to be expected with only 4 points per block and strong spatial dependence.

4.2.2 Dispersion variance of 3 x 3 blocks

The next larger square block is 3 x 3; we repeat the procedure from the previous section, looping over the same number (144) of non-adjacent 3 x 3 blocks in the upper-left quadrant; this is of course a larger total area, but using the same number gives similar statistics for the distribution of the dispersion variance at the different supports.

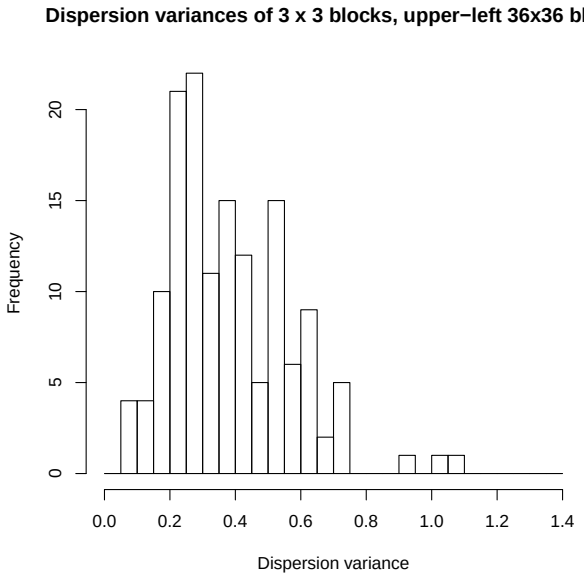
The set of coordinates in each dimension has three values; the looping is by three in each dimension; and the variogram has five bins, at separations $1, \sqrt{2}, 2, \sqrt{1+2^2} = \sqrt{5}, \sqrt{2^2+2^2} = \sqrt{8}$, defined by the straight-line distances between the cell centres.

```
> dv.3 <- vector(mode = "numeric", length = 12^2)
> i <- 1
> for (x in seq(0.5, 34.5, by = 3)) {
+     for (y in seq(0.5, 34.5, by = 3)) {
```

```
+         block <- subset(k.e8.128.pts, (coordinates(k.e8.128.pts)[,
+             1] %in% c(x, x + 1, x + 2)) & (coordinates(k.e8.128.pts)[,
+             2] %in% c(y, y + 1, y + 2)))
+         v <- variogram(z ~ 1, loc = block, cutoff = 2 *
+             sqrt(2))
+         dv.3[i] <- sum(v$np * v$gamma)/sum(v$np)
+         i <- i + 1
+     }
+ }
```

We again summarize the dispersion variances on this support, and display as a histogram:

```
> summary(dv.3)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0751  0.2440  0.3490  0.3810  0.5120  1.0700

> hist(
+     dv.3,
+     main="Dispersion variances of 3 x 3 blocks, upper-left 36x36 block",
+     xlab="Dispersion variance", breaks=seq(0,1.4,by=.05))
```



The average dispersion variance at this support is shown in the summary as 0.3808.

Q5 : *How does the dispersion variance change with increasing support, from 2 x 2 to 3 x 3 blocks? Explain why.* *Jump to A5* •

4.2.3 Dispersion variance of 4 x 4 blocks

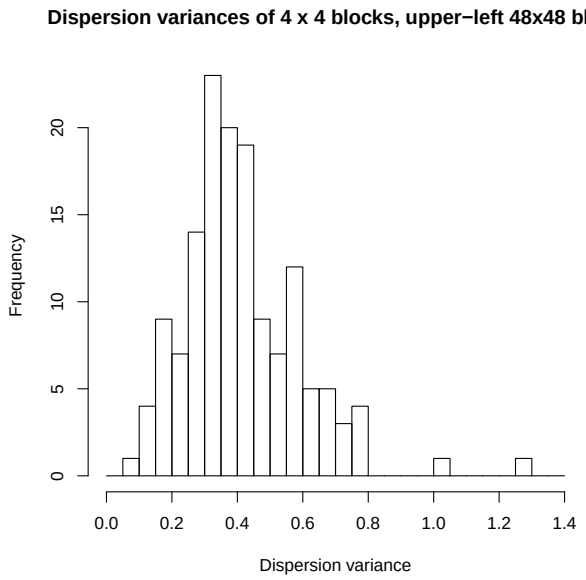
We now increase the size of the block to 4 x 4.

Task 7 : Compute and summarize the dispersion variance at 4 x 4 block support. •

The set of coordinates in each dimension has four values; the looping is by four in each dimension; and the variogram has nine bins, at separations $1, 2, 3, \sqrt{2}, \sqrt{1+2^2} = \sqrt{5}, \sqrt{2^2+2^2} = \sqrt{8}, \sqrt{1+3^2} = \sqrt{10}, \sqrt{2^2+3^2} = \sqrt{13}$, and $\sqrt{3^2+3^2} = \sqrt{18}$, defined by the straight-line distances between the cell centres.

Loop over the same number (144) of non-adjacent 4 x 4 blocks in the upper-left quadrant:

```
> dv.4 <- vector(mode = "numeric", length = 12^2)
> i <- 1
> for (x in seq(0.5, 45.5, by = 4)) {
+     for (y in seq(0.5, 45.5, by = 4)) {
+         block <- subset(k.e8.128.pts, (coordinates(k.e8.128.pts)[,
+             1] %in% c(x, x + 1, x + 2, x + 3)) & (coordinates(k.e8.128.pts)[,
+             2] %in% c(y, y + 1, y + 2, y + 3)))
+         v <- variogram(z ~ 1, loc = block, cutoff = 4 *
+             sqrt(2), width = 0.1)
+         dv.4[i] <- sum(v$np * v$gamma)/sum(v$np)
+         i <- i + 1
+     }
+ }
> summary(dv.4)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0959  0.3020  0.3840  0.4150  0.5170  1.2500

> hist(
+     dv.4,
+     main="Dispersion variances of 4 x 4 blocks, upper-left 48x48 block",
+     xlab="Dispersion variance", breaks=seq(0,1.4,by=.05))
```

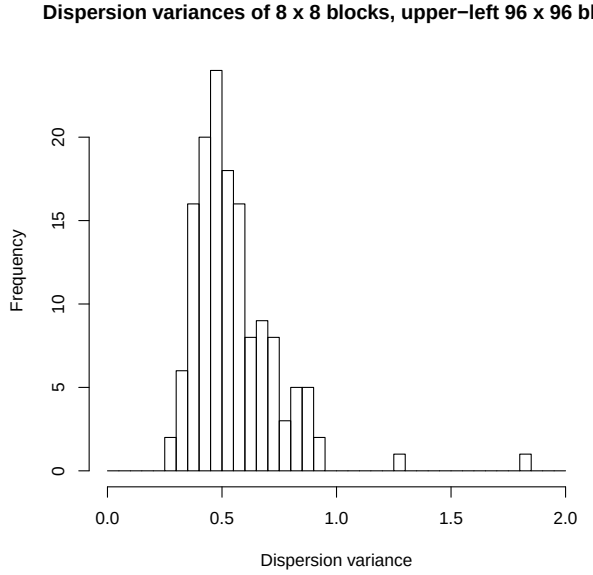


Q6 : *How does the dispersion variance change with increasing support from 3 x 3 to 4 x 4 blocks?* *Jump to A6* •

4.2.4 Dispersion variance of 8 x 8 blocks

Finally, we double the size of the block to 8 x 8, using non-adjacent 8 x 8 blocks in the upper-left quadrant (now, 96 x 96):

```
> dv.8 <- vector(mode = "numeric", length = 12^2)
> i <- 1
> for (x in seq(0.5, 88.5, by = 8)) {
+     for (y in seq(0.5, 88.5, by = 8)) {
+         block <- subset(k.e8.128.pts, (coordinates(k.e8.128.pts)[,
+             1] %in% c(x, x + 1, x + 2, x + 3, x + 4,
+             x + 5, x + 6, x + 7)) & (coordinates(k.e8.128.pts)[,
+             2] %in% c(y, y + 1, y + 2, y + 3, y + 4,
+             y + 5, y + 6, y + 7)))
+         v <- variogram(z ~ 1, loc = block, cutoff = 8 *
+             sqrt(2), width = 0.1)
+         dv.8[i] <- sum(v$np * v$gamma)/sum(v$np)
+         i <- i + 1
+     }
+ }
> summary(dv.8)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.283   0.435   0.511   0.552   0.623   1.800

> hist(
+     dv.8,
+     main="Dispersion variances of 8 x 8 blocks, upper-left 96 x 96 block",
+     xlab="Dispersion variance", breaks=seq(0,2,by=.05))
```

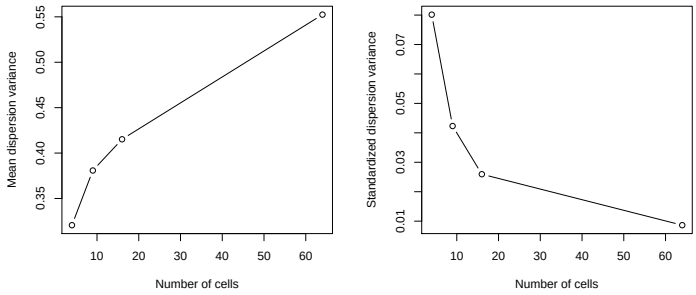


4.2.5 Change of dispersion variance with support

We've now computed dispersion variances for 2 x 2, 3 x 3, 4 x 4, and 8 x 8 supports. This allows us to examine the trend of dispersion variance with increasing support size. However, the supports have different numbers of cells, so it is also interesting to see the mean dispersion variance per cell.

Task 8 : Compare the means, and also the mean standardized by number of cells in the support. •

```
> dv.means <- rbind(`2x2` = mean(dv.2), `3x3` = mean(dv.3),
+     `4x4` = mean(dv.4), `8x8` = mean(dv.8))
> dv.means.cells <- c(2^2, 3^2, 4^2, 8^2)
> dv.means.std <- dv.means/dv.means.cells
> data.frame(means = dv.means, `std means` = dv.means.std)

      means std.means
2x2 0.32060 0.0801508
3x3 0.38082 0.0423134
4x4 0.41518 0.0259489
8x8 0.55243 0.0086317

> par(mfrow = c(1, 2))
> plot(dv.means ~ dv.means.cells, type = "b", main = "",
+     xlab = "Number of cells", ylab = "Mean dispersion variance")
> plot(dv.means.std ~ dv.means.cells, type = "b", main = "",
+     xlab = "Number of cells", ylab = "Standardized dispersion variance")
> par(mfrow = c(1, 2))
```

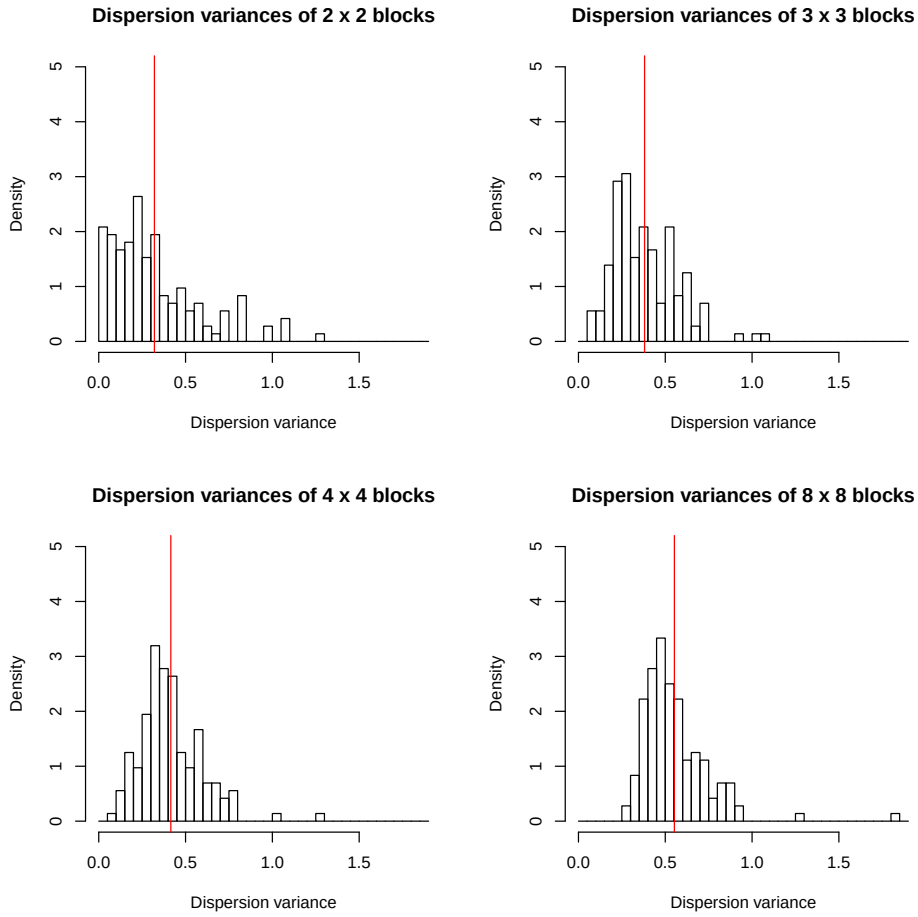


Q7 : *What is the trend of the mean dispersion variance with increasing with increasing support, from 2 x 2 through 8 x 8 cells?* *Jump to A7* •

Task 9 : Compare the histograms of the dispersion variances, on the same scale. •

We also plot the mean value as a vertical red bar. Note the use of the `freq` optional argument to specify that we want a density histogram (`freq=FALSE`) rather than the actual counts of cells; this allows us to compare the distributions on the same scale.

```
> par(mfrow = c(2, 2))
> hist(dv.2, freq = F, main = "Dispersion variances of 2 x 2 blocks",
+     xlab = "Dispersion variance", breaks = seq(0, 1.9,
+         by = 0.05), ylim = c(0, 5))
> abline(v = mean(dv.2), col = "red")
> hist(dv.3, freq = F, main = "Dispersion variances of 3 x 3 blocks",
+     xlab = "Dispersion variance", breaks = seq(0, 1.9,
+         by = 0.05), ylim = c(0, 5))
> abline(v = mean(dv.3), col = "red")
> hist(dv.4, freq = F, main = "Dispersion variances of 4 x 4 blocks",
+     xlab = "Dispersion variance", breaks = seq(0, 1.9,
+         by = 0.05), ylim = c(0, 5))
> abline(v = mean(dv.4), col = "red")
> hist(dv.8, freq = F, main = "Dispersion variances of 8 x 8 blocks",
+     xlab = "Dispersion variance", breaks = seq(0, 1.9,
+         by = 0.05), ylim = c(0, 5))
> abline(v = mean(dv.8), col = "red")
> par(mfrow = c(1, 1))
```



Q8 : *How does the distribution of the dispersion variance change with increasing support, from 2 x 2 through 8 x 8 cells?* *Jump to A8* •

Challenge: Compute the dispersion variances for some other block sizes (e.g. 5x5, 6x6, 12x12) and re-create the graph of mean variance vs. block size including these sizes.

5 Effect of support

In this section we investigate how change of support affects summary statistics and variograms. This will then lead to methods to change support.

Task 10 : Create three grids of increasingly-coarse resolution, with the same bounding box as the 128x128 grid created in §4. •

We create the grid topologies with `GridTopology` and use these to build a spatial grid with `SpatialGrid`. The trick here is to specify the grid resolution with the second argument (`cellsize`) so that this, multiplied by the cell dimension, specified with the third argument (`cells.dim`), are the same

as the original grid, i.e. 128. The centre of the first cell (first argument, `cellcentre.offset`) must also be moved to the centre of the cell.

```
> grid.64 <- SpatialGrid(GridTopology(cellcentre.offset = c(2,
+     2), cellsize = c(4, 4), cells.dim = c(32, 32)))
> grid.32 <- SpatialGrid(GridTopology(cellcentre.offset = c(4,
+     4), cellsize = c(8, 8), cells.dim = c(16, 16)))
> grid.16 <- SpatialGrid(GridTopology(cellcentre.offset = c(8,
+     8), cellsize = c(16, 16), cells.dim = c(8, 8)))
> summary(grid.128)

Object of class SpatialGrid
Coordinates:
     min max
[1,]   0 128
[2,]   0 128
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
1               0.5        1       128
2               0.5        1       128

> summary(grid.64)

Object of class SpatialGrid
Coordinates:
     min max
[1,]   0 128
[2,]   0 128
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
1                 2        4        32
2                 2        4        32

> summary(grid.32)

Object of class SpatialGrid
Coordinates:
     min max
[1,]   0 128
[2,]   0 128
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
1                 4        8        16
2                 4        8        16

> summary(grid.16)

Object of class SpatialGrid
Coordinates:
     min max
```

```
[1,]   0 128
[2,]   0 128
Is projected: NA
proj4string : [NA]
Grid attributes:
  cellcentre.offset cellsize cells.dim
1                 8       16         8
2                 8       16         8
```

Note that these all have the same bounding box (minimum and maximum coördinates) but different numbers of cells, different cell size, and therefore different offsets to the cell centre from the grid edge.

Task 11 : Aggregate the data values from the 128 x 128 grid into increasingly-coarser grids. •

It may help your understanding to visualize this example as a satellite image of different resolutions, where the digital numbers (radiances) of the coarser-resolution pixels are averages of a square of finer-resolution pixels; this is the image resulting from physical integration, i.e. one sensor capturing energy from a larger area (coarse resolution) which could also be sensed by an array of sensors each capturing energy from a smaller area (fine resolution). Here we use mathemetical integration (averaging) to simulate physical integration (larger sensor field of view).

We have the three coarser grids, with the same bounding box as the fine grid. Values of the target variable can be computed for each grid cell using the `idw` "Inverse-Distance Weighted" method, with zero-power (i.e. all neighbours are weighted the same, no distance decay) specified with the `idp` "inverse distance power" argument, and the required number of neighbours from the fine grid, specified with the `nmax` argument. For the 64 x 64 grid, this is 4 (i.e. four cells are aggregated into one, halving the resolution); for the 32 x 32 grid, this is 16; and for the 16 x 16 grid, this is 32 cells into one.

```
> k.e8.64 <- idw(z ~ 1, loc = k.e8.128, newdata = grid.64,
+     idp = 0, nmax = 4)

[inverse distance weighted interpolation]

> names(k.e8.64) <- c("z", "nr")
> k.e8.32 <- idw(z ~ 1, loc = k.e8.128, newdata = grid.32,
+     idp = 0, nmax = 16)

[inverse distance weighted interpolation]

> names(k.e8.32) <- c("z", "nr")
> k.e8.16 <- idw(z ~ 1, loc = k.e8.128, newdata = grid.16,
+     idp = 0, nmax = 32)

[inverse distance weighted interpolation]

> names(k.e8.16) <- c("z", "nr")
```

Note: This series of grids could also be computed (within rounding error) stepwise, with each coarser grid being aggregated from the next-finer resolution, always with 4 neighbours. For example, the 64x64 grid could be aggregated from the 128x128 grid, using the four nearest neighbours, i.e., a 2 x 2 grid at 128x128 is aggregated into one cell at 64x64.

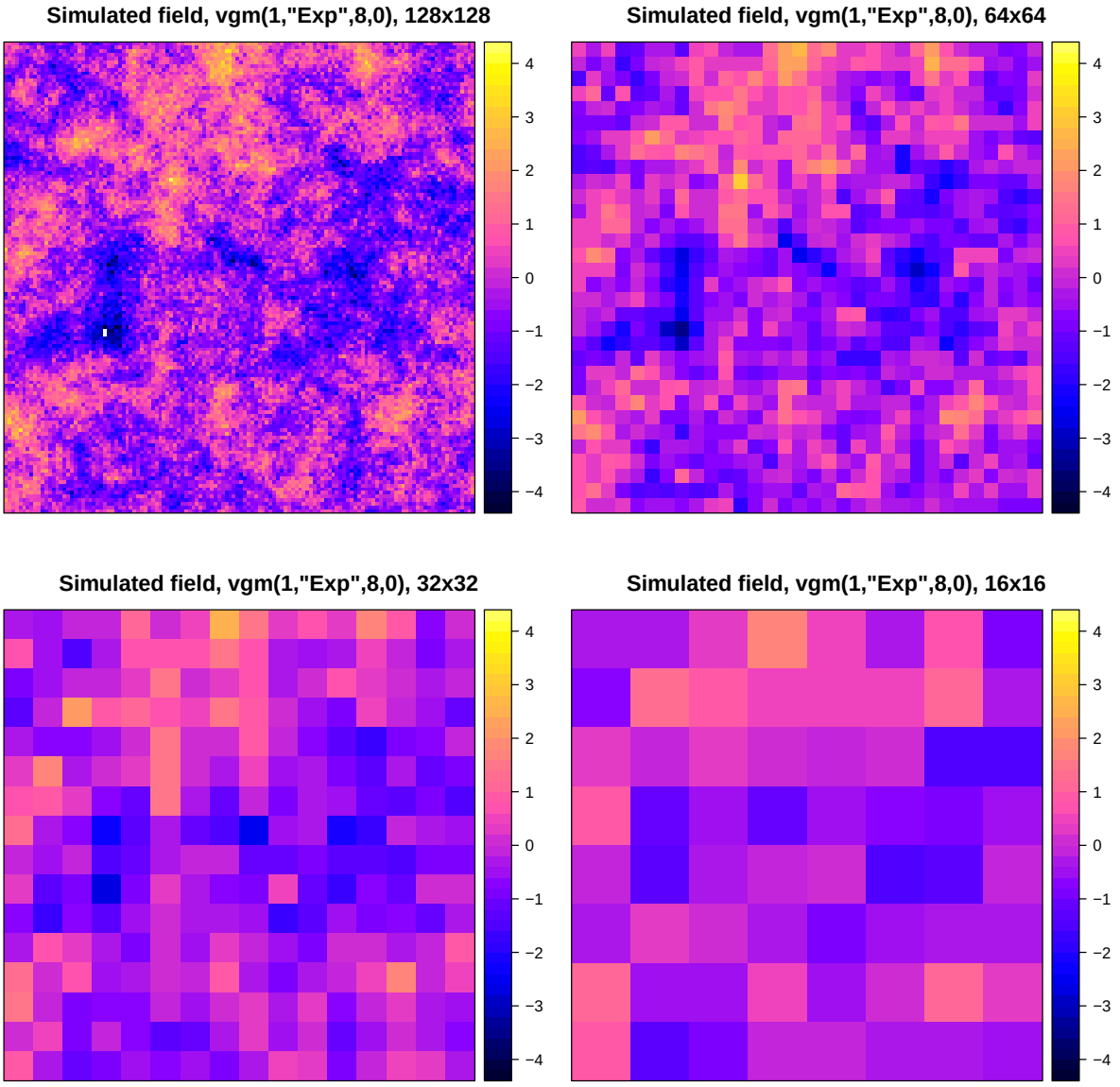
```
> tmp <- idw(z ~ 1, loc = k.e8.128, newdata = grid.64,
+     idp = 0, nmax = 4)

[inverse distance weighted interpolation]

> summary(tmp$var1.pred)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.460  -0.809  -0.271  -0.230   0.329   3.030

> summary(k.e8.64$z)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.460  -0.809  -0.271  -0.230   0.329   3.030

> rm(tmp)
```

The summary statistics are identical.

Task 12 : Display the spatial fields at the four aggregation levels on the same plot. •

```
> z.at <- seq(-4.4, 4.4, by = 0.2)
> sim.plot.128 <- spplot(k.e8.128, zcol = "z", col.regions = bpy.colors(64),
+     main = "Simulated field, vgm(1,\"Exp\",8,0), 128x128",
+     at = z.at)
> sim.plot.64 <- spplot(k.e8.64, zcol = "z", col.regions = bpy.colors(64),
+     main = "Simulated field, vgm(1,\"Exp\",8,0), 64x64",
+     at = z.at)
> sim.plot.32 <- spplot(k.e8.32, zcol = "z", col.regions = bpy.colors(64),
+     main = "Simulated field, vgm(1,\"Exp\",8,0), 32x32",
+     at = z.at)
> sim.plot.16 <- spplot(k.e8.16, zcol = "z", col.regions = bpy.colors(64),
+     main = "Simulated field, vgm(1,\"Exp\",8,0), 16x16",
+     at = z.at)
```

```
> print(sim.plot.128, split = c(1, 1, 2, 2), more = T)
> print(sim.plot.64, split = c(2, 1, 2, 2), more = T)
> print(sim.plot.32, split = c(1, 2, 2, 2), more = T)
> print(sim.plot.16, split = c(2, 2, 2, 2), more = F)
```



Q9 : *How does the spatial field change as aggregation level is increased?*
Jump to A9 •

5.1 Effect of support on non-spatial statistics

Task 13 : Compare summary statistics for the four levels of aggregation, including the naïve variances. •

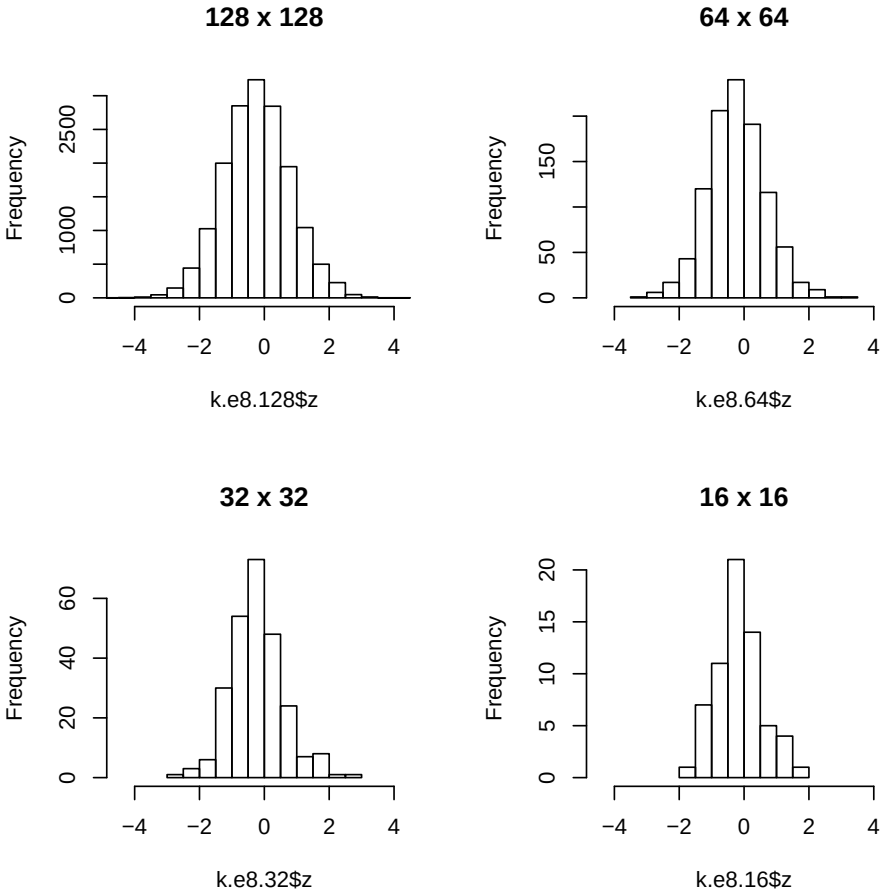
To display these as a nice table, we first stack the rows of the summary vectors using `rbind` "row bind", and then add another column to the table with `cbind` "column bind"; this column is a vector of the variances, created with the `c` "catenate" function.

```
> cbind(rbind(summary(k.e8.128$z),
+             summary(k.e8.64$z),
+             summary(k.e8.32$z),
+             summary(k.e8.16$z)),
+       "Var."=c(
+           var(k.e8.128$z),
+           var(k.e8.64$z),
+           var(k.e8.32$z),
+           var(k.e8.16$z)))

      Min. 1st Qu. Median   Mean 3rd Qu. Max.    Var.
[1,] -5.00  -0.914 -0.239 -0.230   0.441 4.40 1.04044
[2,] -3.46  -0.809 -0.271 -0.230   0.329 3.03 0.78204
[3,] -2.63  -0.798 -0.256 -0.233   0.231 2.55 0.65752
[4,] -1.57  -0.553 -0.235 -0.147   0.322 1.63 0.51671
```

Q10 : *How does aggregation level affect the univariate distributions of the simulated variable?* *Jump to A10* •

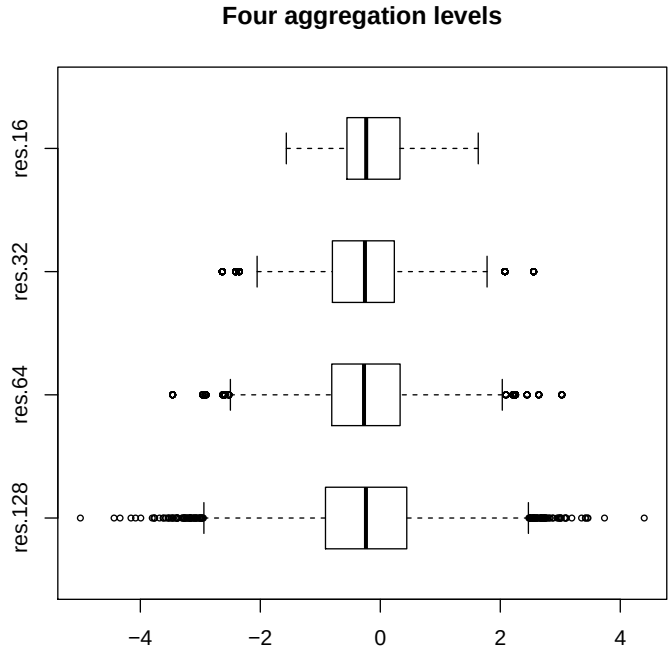
Task 14 : Compare the histograms and boxplots for the four levels of aggregation. •

```
> par(mfrow = c(2, 2))
> hist(k.e8.128$z, xlim = c(-4.5, 4.5), main = "128 x 128")
> hist(k.e8.64$z, xlim = c(-4.5, 4.5), main = "64 x 64")
> hist(k.e8.32$z, xlim = c(-4.5, 4.5), main = "32 x 32")
> hist(k.e8.16$z, xlim = c(-4.5, 4.5), main = "16 x 16")
> par(mfrow = c(1, 1))
```



To produce a single boxplot with the four aggregation levels, we create a data frame, using the `data.frame` function, with one named field per level, and pass this data frame to `boxplot`.

```
> boxplot(data.frame(res.128=k.e8.128$z,
+                    res.64=k.e8.64$z,
+                    res.32=k.e8.32$z,
+                    res.16=k.e8.16$z),
+         main="Four aggregation levels",
+         horizontal=T, boxwex=.5, cex=.6)
```



Q11 : *How does aggregation level affect (1) the summary statistics; (2) the naïve variances; (3) the univariate distributions of the simulated variable?* *Jump to A11* •

5.2 Effect of support on spatial statistics

The coarse random fields of §5 have spatial structure, because they were constructed from the known structure at the sampling support. How is this structure affected by aggregation?

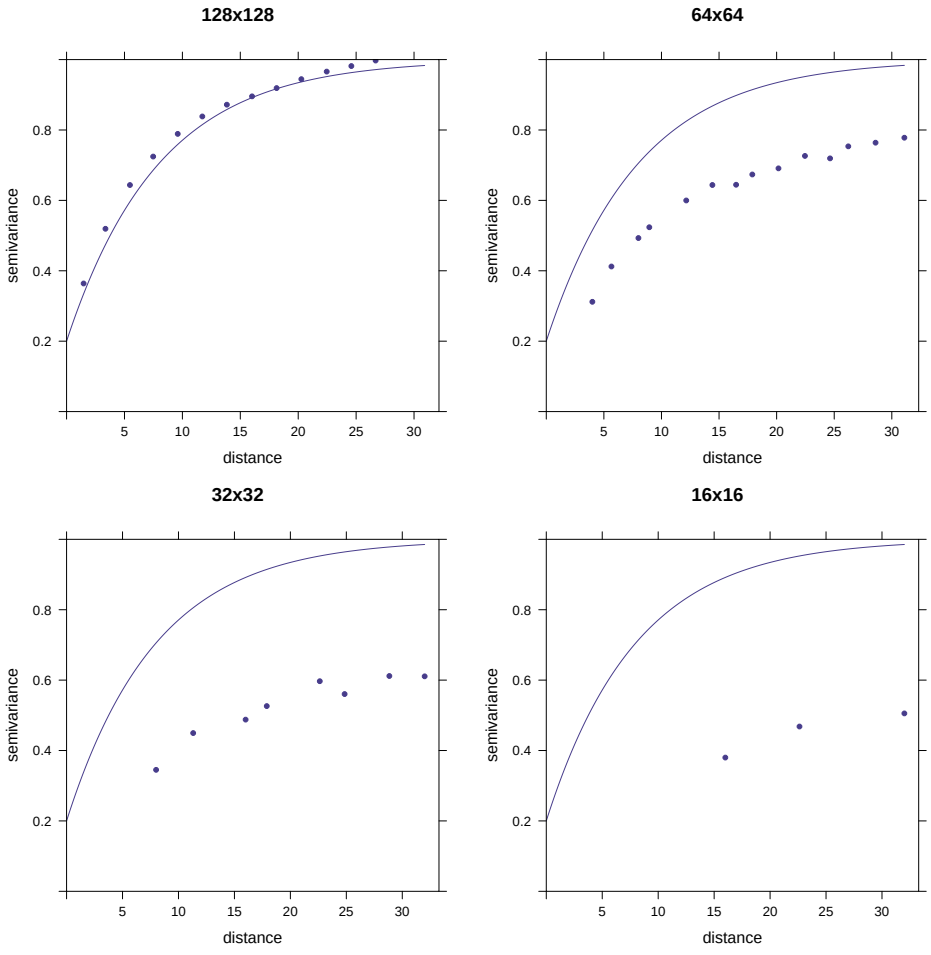
Task 15 : Compare the variograms from the four levels; with the known model (used to create the finest-resolution field) superimposed. •

We use the `variogram` function to compute the empirical variogram out to a cutoff 1.5 times the known effective range of 24. Since we only used this range in the simulation, we expect that the variogram at larger separation will be flatter (lower than) the model.

> **Note:** To visualize the variogram, we change the `lattice` graphics parameters, to get a darker-blue, filled dots. The named colour is one of the 657 fixed colours given by the `colors` function. The `simpleTheme` function is an easy way to store a list of the basic plot parameters.
>
> The `trellis.par.set`.function changes the graphics parameters. Since each figure in this document is a separate PDF, each time these new parameters should be used in a `lattice` graph, they must be specified with `trellis.par.set`.

More complicated manipulations of the `lattice` graphics paramters require that the old parameters be retrieved with `trellis.par.get`, changed with regular assignment statements, and re-written with `trellis.par.set`.

```
> v.128 <- variogram(z ~ 1, loc = k.e8.128, cutoff = 32)
> v.64 <- variogram(z ~ 1, loc = k.e8.64, cutoff = 32)
> v.32 <- variogram(z ~ 1, loc = k.e8.32, cutoff = 32)
> v.16 <- variogram(z ~ 1, loc = k.e8.16, cutoff = 32)
> myTheme <- simpleTheme(col = "slateblue4", pch = 20,
+     cex = 0.9)
> v.128.pl <- plot(v.128, ylim = c(0, 1), pl = F, model = vm,
+     main = "128x128")
> v.64.pl <- plot(v.64, ylim = c(0, 1), pl = F, model = vm,
+     main = "64x64")
> v.32.pl <- plot(v.32, ylim = c(0, 1), pl = F, model = vm,
+     main = "32x32")
> v.16.pl <- plot(v.16, ylim = c(0, 1), pl = F, model = vm,
+     main = "16x16")

> trellis.par.set(myTheme)
> print(v.128.pl, split = c(1, 1, 2, 2), more = T)
> print(v.64.pl, split = c(2, 1, 2, 2), more = T)
> print(v.32.pl, split = c(1, 2, 2, 2), more = T)
> print(v.16.pl, split = c(2, 2, 2, 2), more = F)
```



Q12 : *What happens to the experimental variogram as aggregation increases? How well do these fit the known model?* *Jump to A12*

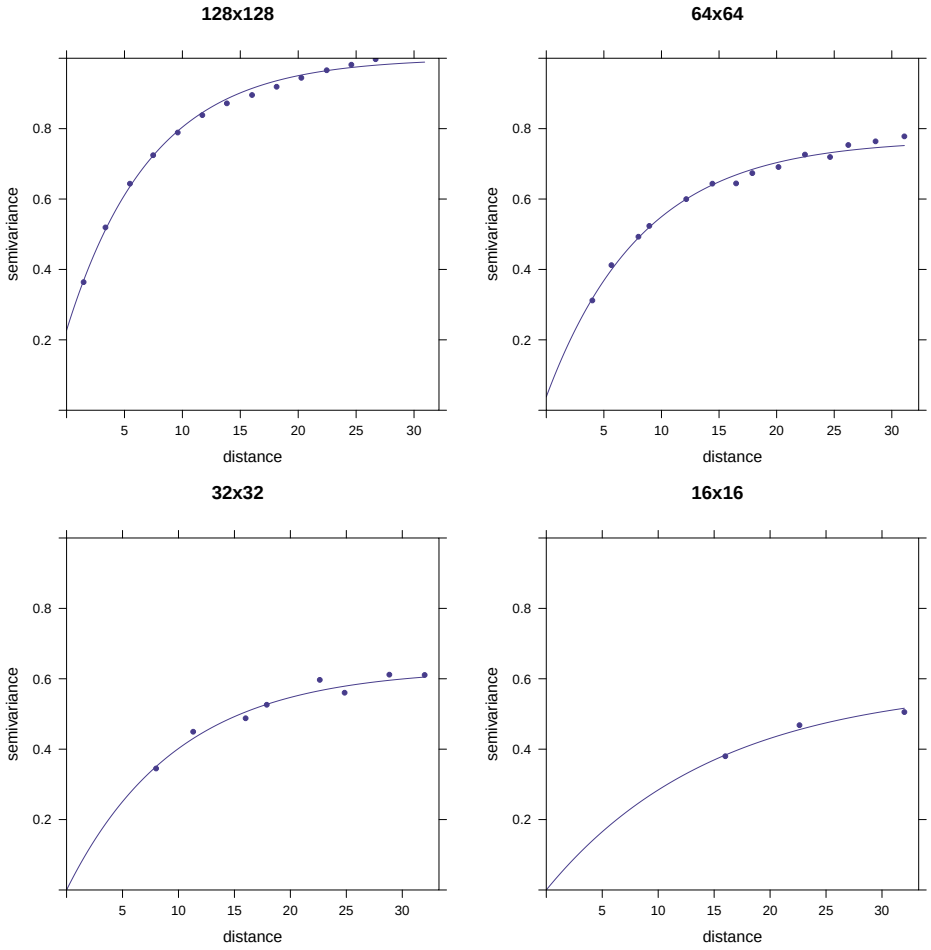
•

Although the parameters change with increased aggregation, the model form should not (the underlying process should be the same). So an exponential model should fit all of them.

Task 16 : Attempt to fit these empirical variograms starting from the known model. •

We use the `fit.variogram` function, with default fitting method, and display the fit.

```
> (vmf.128 <- fit.variogram(v.128, vm))

  model   psill  range
1   Nug 0.22719 0.0000
2   Exp 0.77288 7.2819

> (vmf.64 <- fit.variogram(v.64, vm))

  model    psill  range
1   Nug 0.037907 0.0000
2   Exp 0.731472 8.3157

> (vmf.32 <- fit.variogram(v.32, vm))

  model   psill  range
1   Nug 0.00000 0.0000
2   Exp 0.62903 9.8195

> (vmf.16 <- fit.variogram(v.16, vm))

  model   psill  range
1   Nug 0.00000  0.000
2   Exp 0.58686 15.107

> v.128.pl <- plot(v.128, ylim = c(0, 1), pl = F, model = vmf.128,
+     main = "128x128")
> v.64.pl <- plot(v.64, ylim = c(0, 1), pl = F, model = vmf.64,
+     main = "64x64")
> v.32.pl <- plot(v.32, ylim = c(0, 1), pl = F, model = vmf.32,
+     main = "32x32")
> v.16.pl <- plot(v.16, ylim = c(0, 1), pl = F, model = vmf.16,
+     main = "16x16")

> trellis.par.set(myTheme)
> print(v.128.pl, split = c(1, 1, 2, 2), more = T)
> print(v.64.pl, split = c(2, 1, 2, 2), more = T)
> print(v.32.pl, split = c(1, 2, 2, 2), more = T)
> print(v.16.pl, split = c(2, 2, 2, 2), more = F)
```



Q13 : *Was the automatic fitting method able adjust the parameters? What happens to the variogram parameters with increasing aggregation? How well does this model form fit the variograms?* *Jump to A13* •

6 Variogram regularization

The term **regularization**, applied to variograms, refers to the process by which a variogram at one support is related to that at another.

We first consider going from a punctual to a block support. A practical application is when we have a variogram from punctual support, but the further work will be on a larger support, and we need to estimate the variogram at that support to plan sampling. An example would be a variogram of soil properties of individual soil cores (approximately 10 cm diameter), where further sampling will be by composite sampling from farmers' fields of 1 ha (100 x 100 m), because whatever intervention is planned will be at that support.

Suppose the punctual support is $\boldsymbol{b}$. We refer to the larger support as $\boldsymbol{B}$. The variogram on support $\boldsymbol{B}$ can often be related to that on support $\boldsymbol{b}$ by:

$$\gamma_B(\mathbf{h}) \approx \gamma_b(\mathbf{h}) - \overline{\gamma}(B,B) \tag{13}$$

where $\mathbf{h}$ is the separation vector, here between centroids of the larger support B, and $\overline{\gamma}(B,B)$ is the average semivariance within the support B, i.e. the dispersion variance as explained in §3. In words, the variogram at block support is translated downward from the punctual semivariogram by $\overline{\gamma}(B,B)$.

In addition, the range is automatically increased by the diameter of the larger support.

> **Note:** This approximation only holds when $|\mathbf{h}| >> \sqrt{|B|}$. At shorter ranges the variogram at large support is not just a translation downwards of the punctual variogram; it is concave. This is logical, because at a certain sepration the downward translation would result in a negative nugget, which is not possible.
>
> See [9, §5 & 6] for details of this calculation.

6.1 Computing the dispersion variance within a block

To use this approximation, we need to compute the dispersion variance within the block, i.e. $\overline{\gamma}(B,B)$ of Equation 13. One method is to discretize the block and compute the variogram cloud on a simulated field of that block size, as in §4.2.

Of course, there are an infinity of 0-dimensional points in any block; and even considering a finite point support of one grid cell in this simulated field, there are 16384 supports in 1 ha. How many are necessary to get a satisfactory discretization?

We break this task down into steps:

1. Specify a variogram model – we use the one that was used to simulate the field, i.e., exponential with partial sill 0.8, nugget 0.2, range parameter 8 so effective range 24;
2. Simulate a field with this model over the block size – we use the 128 x 128 field already simulated;
3. Select points from the field, at increasingly-fine discretizations;
4. Compute the dispersion variances;
5. Determine how fine a discretization is needed to approximate the "infinite" dispersion variance.

Task 17 : Create the variogram model as specified above. •

```
> vm <- vgm(psill = 0.8, , model = "Exp", range = 8, nugget = 0.2)
```

Task 18 : Considering the 128 x 128 grid of §4.1 as the block, compute the dispersion variance within this block, from point support with a known punctual variogram, at a 4 x 4 discretization. •

The aim is to see how fine a discretization is necessary to accurately compute the dispersion variance. We will try increasingly-fine discretizations. We first illustrate the procedure with a 4 x 4 grid of points, regularly spaced.

Task 19 : Create sampling points, on a regular grid within this block, of $2^4 = 16$ points. •

An interesting way to get a regular grid is to use **k-means clustering** to achieve a **spatial coverage sample**. This is explained briefly by Walvoort et al. [12] and more completely by de Gruijter et al. [2, §8.3.3]. These authors have prepared an R package `spcosa` [13].

The simplest method is to minimize the mean squared shortest distance (MSSD) from the sample to the grid nodes. Considering N points indexed by i, and the grid as a set of points indexed by j:

$$\frac{1}{N}\sum_{i}^{N}\min_{j}\left(D_{ij}^2\right) \tag{14}$$

This can be minimized by k-means algorithm on the discretization grid, resulting in a set of cluster centroids. All we need to do is to specify the number of points and their coordinates.

> **Note:** Note that if the study area is not convex, the resulting centroid is not restricted to the study area; it may be that the maximum information is found by sampling outside.

A simple approach is to use the `kmeans` function with the default value of the `algorithm` argument, i.e, `Hartigan-Wong` [4]. This partitions the points into a user-specified number (referred to as "k") groups, to mimimize the sum of squares from points to the assigned cluster centres. This is best-known in feature (multivariate) space, but there is no reason it can't be applied in geographic space, with the coordinates as the multi-variables. In this case the points are centroids of Theissen polygons.

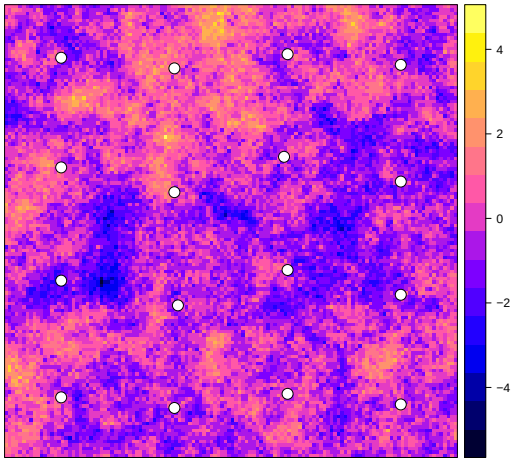
Again, we use `set.seed` so that your results will match these notes; we specify 16 groups, i.e., 16 cluster centres:

```
> set.seed(316318)
> scheme <- data.frame(round(kmeans(x=coordinates(grid.128),centers=2^4,
+                                   iter.max=1000,nstart=2^4,
+                                   algorithm="Hartigan-Wong")$centers))
> names(scheme) <- c("x","y"); coordinates(scheme) <- ~x + y;
> pts.z <- overlay(k.e8.128, scheme)
> summary(pts.z)

Object of class SpatialPointsDataFrame
Coordinates:
  min max
x  16 112
y  14 114
Is projected: NA
```

```
proj4string : [NA]
Number of points: 16
Data attributes:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -2.310  -0.431  -0.295  -0.263   0.179   0.809
```

Task 20 : Plot the block and the discretization points. •

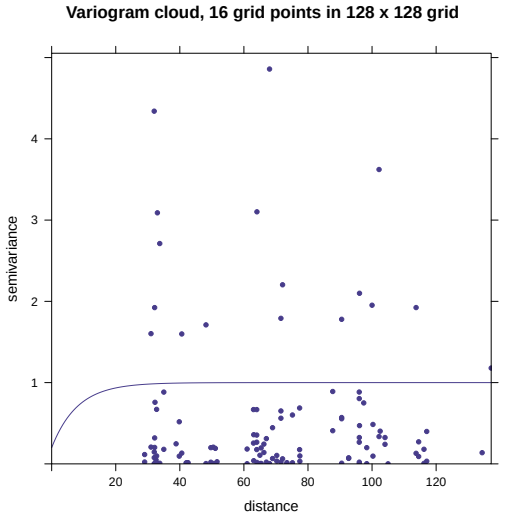
```
> layout.2 <- list("sp.points", pts.z, pch = 21, cex = 1.5,
+     col = "black", fill = "white")
> print(spplot(k.e8.128, col.regions = bpy.colors(64),
+     sp.layout = list(layout.2)))
```



Task 21 : Compute the dispersion variance of this set of points from the variogram cloud. Plot the cloud with all point-pairs. •

The variogram cloud is computed with `variogram`, with the `cloud=T` optional argument; the dispersion variance is the mean of the point-pair semivariances. To show all the point-pairs, we must specify the `cutoff` argument to `variogram`; by default it is 1/3 the maximum separation. The diagonal distance across the grid is $128\sqrt{2}$.

```
> vc <- variogram(z ~ 1, data=pts.z, cloud=T,
+                 cutoff=128*sqrt(2))
> mean(vc$gamma)

[1] 0.5329

> trellis.par.set(myTheme)
> print(plot(
+     vc,
+     main="Variogram cloud, 16 grid points in 128 x 128 grid",
+     model=vm))
```



Q14 : *What are the main features of the variogram cloud at this discretization?* *Jump to A14*

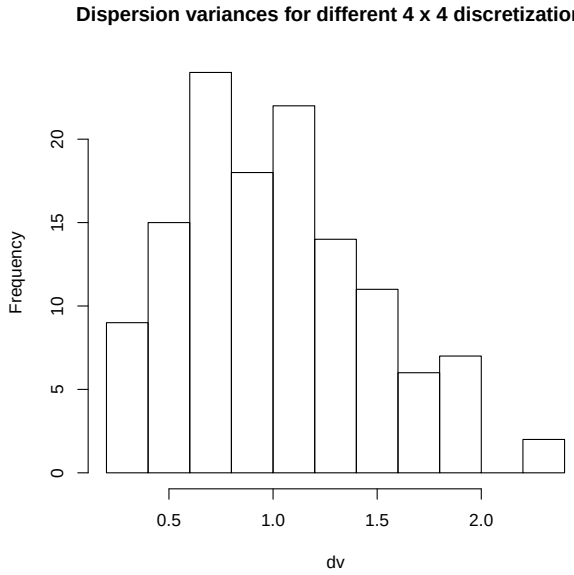
•

Of course, different discretizations (selection of points) will give different dispersion variances; how big is this effect?

Task 22 : Compute the dispersion variance for several different 4 x 4 discretizations; summarize the statistics. •

For this we use the `spsample` method to place points on a grid (or within a polygon); if used with the `"regular"` `type` parameter, it selects a random starting point and then creates a regular grid.

```
> set.seed(1768)
> dv <- vector(mode="numeric", length=128)
> for (i in 1:128) {
+   pts <- spsample(k.e8.128, 4^2, type="regular")
+   pts.z <- overlay(k.e8.128, pts)
+   dv[i] <- mean(variogram(z ~ 1, data=pts.z,
+                           cutoff=128*sqrt(2),
+                           cloud=T)$gamma)
+ }
> summary(dv)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.325   0.673   0.953   1.020   1.290   2.280

> hist(
+     dv,
+     main="Dispersion variances for different 4 x 4 discretizations")
> print(paste("Naive variance:",
+             round(var(k.e8.128$z),4)))

[1] "Naive variance: 1.0404"
```



Q15 : *How much variability is there between these different discretizations? Is the 4 x 4 discretization a reliable estimate of the dispersion variance?* *Jump to A15* •

6.1.1 Comparing dispersion variance of 4 x 4 . . . 16x16 points

We now compute dispersion variances for various sizes and collect into an array to see the trend. In place of the variogram cloud we use the averaged variogram with narrow bins, since the larger grids have a large number of point-pairs[1].

Initialize the results array:

```
> dv <- matrix(0, nrow = 128, ncol = 6)
> colnames(dv) <- c("4x4", "6x6", "8x8", "10x10", "12x12",
+     "16x16")
```

Now repeat the computation of dispersion variance for each of these discretizations, 128 times each, and review the summary statistics:

```
> set.seed(1814)
> n <- c(4^2, 6^2, 8^2, 10^2, 12^2, 16^2)
> for (p in 1:6) {
+     for (i in 1:128) {
+         pts <- spsample(k.e8.128, n[p], type = "regular")
+         pts.z <- overlay(k.e8.128, pts)
+         v <- variogram(z ~ 1, data = pts.z, width = 1,
+             cutoff = 128 * sqrt(2))
+         dv[i, p] <- sum(v$np * v$gamma)/sum(v$np)
+     }
```

[1] e.g. the 10x10 grid has $(100 \times 99)/2 = 4950$ pairs

```
+ }
> summary(dv)

      4x4              6x6              8x8             10x10
 Min.   :0.289   Min.   :0.440   Min.   :0.707   Min.   :0.686
 1st Qu.:0.734   1st Qu.:0.843   1st Qu.:0.921   1st Qu.:0.948
 Median :0.984   Median :1.004   Median :1.076   Median :1.057
 Mean   :1.060   Mean   :1.021   Mean   :1.068   Mean   :1.055
 3rd Qu.:1.286   3rd Qu.:1.160   3rd Qu.:1.207   3rd Qu.:1.153
 Max.   :2.220   Max.   :1.789   Max.   :1.564   Max.   :1.433
     12x12           16x16
 Min.   :0.807   Min.   :0.886
 1st Qu.:0.945   1st Qu.:0.992
 Median :1.032   Median :1.043
 Mean   :1.042   Mean   :1.045
 3rd Qu.:1.137   3rd Qu.:1.089
 Max.   :1.389   Max.   :1.193

> apply(dv, 2, sd)

     4x4      6x6      8x8    10x10    12x12    16x16
0.422318 0.244965 0.176287 0.151172 0.115838 0.067696

> apply(dv, 2, function(x) 100 * sd(x)/mean(x))

    4x4     6x6     8x8   10x10   12x12   16x16
39.8225 23.9951 16.5083 14.3340 11.1223  6.4759
```

The last computation is the coefficient of variation (CV%).

Q16 : *(1) What is the trend of the mean dispersion variance as the discretization level increases? (2) What is the trend of the range and standard deviation of different discretizations at the same level of detail?* *Jump to A16* •

Q17 : *So, what is an acceptable discretization for this 128 x 128 block and variogram with effective range 24? What proportion of the points is this?* *Jump to A17* •

6.2 Computing the dispersion variance from the variogram

In practice we do not know the spatial structure of the presumed random field; all we have is a single empirical variogram from the punctual sampling. We can use this to estimate the within-block semivariance $\overline{\gamma}(B, B)$ from the short-range portion of the fitted variogram model.

For example, consider the model used to create the simulated 128 x 128 field, and one realization of an empirical variogram. Sampling to establish the (unknown) spatial structure is often by means of a nested design [15]; there are many other approaches, e.g. [6, 8]. Here we just use a random sample (to ensure many different spacings) on a small portion of the grid.

For this we need to know the approximate range of spatial dependence; we suppose that is known from field experience.

Task 23 : Simulate a random sample of 128 points on a 64 x 64 window of the 128 x 128 simulated field; compute and model the empirical semivariogram; compare to the known variogram model. •

To window a spatial grid, simply subscript by grid row and column, as if the object were a matrix. For example, to extract the upper-left 64 x 64 window:

```
> bbox(k.e8.128)

   min max
s1   0 128
s2   0 128

> k.e8.64 <- k.e8.128[1:64, 1:64]
> bbox(k.e8.64)

   min max
s1   0  64
s2  64 128
```

Note the reduced bounding box.

We use the `spsample` with the `"random"` `type` parameter:

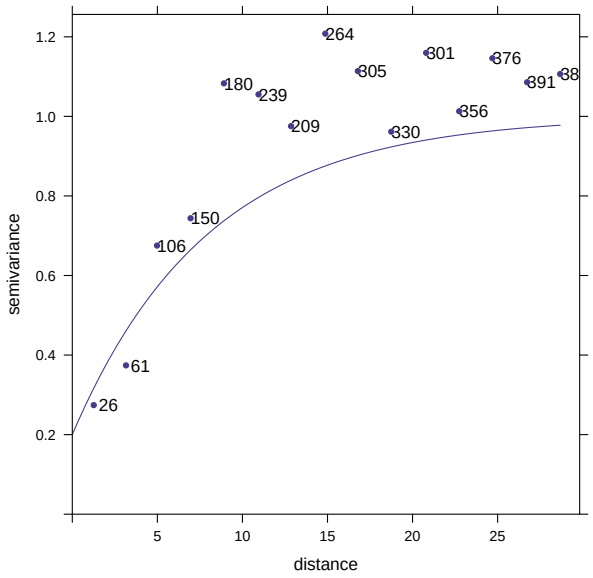
```
> set.seed(250)
> repeat {
+     try(scheme <- spsample(k.e8.64, n = 128, type = "random"),
+         silent = T)
+     if (class(.Last.value) != "try-error")
+         break
+ }
> plot(coordinates(scheme))
> grid()
```

Now extract the data values at these points with `overlay`:

```
> pts.z <- overlay(k.e8.128, scheme)
> summary(pts.z)

Object of class SpatialPointsDataFrame
Coordinates:
        min     max
s1  0.57376  63.677
s2 64.78434 127.435
Is projected: NA
proj4string : [NA]
Number of points: 128
Data attributes:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.8300 -0.6840  0.0438  0.0780  0.8000  3.1900
```

Compute and plot the empirical variogram; superimpose the known model:

```
> v.e <- variogram(z ~ 1, pts.z)
> print(plot(v.e, pl = T, model = vm))
```

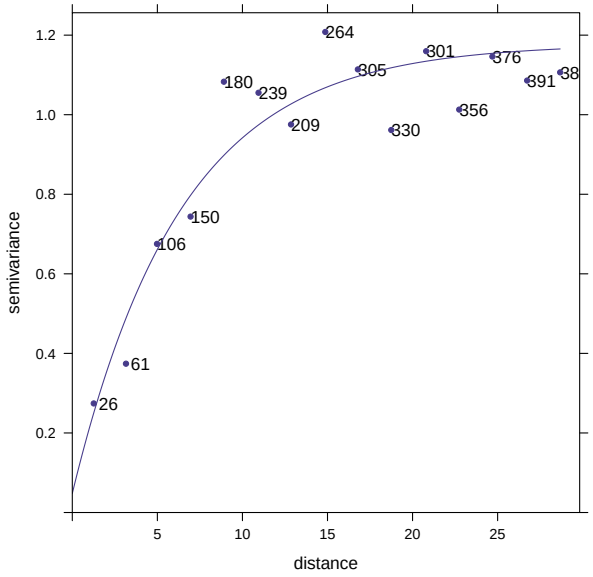


Q18 : *How well does this empirical variogram match the known model, according to visual evidence?* *Jump to A18* •

Of course, if this were a real sample, we would not know the variogram model.

Model the empirical variogram, starting from an eyeball fit:

```
> print(vm)

  model psill range
1   Nug   0.2     0
2   Exp   0.8     8

> (v.emf <- fit.variogram(v.e, model = vgm(1, "Exp", 15/3,
+     0.2)))

  model    psill  range
1   Nug 0.047925 0.0000
2   Exp 1.130032 6.3868

> print(plot(v.e, pl = T, model = v.emf))
```



Q19 : *How well does this empirical variogram match the known model, according to the fit?* *Jump to A19* •

We can assume $|\mathbf{h}| >> \sqrt{|B|}$, so by the relation above, we need to determine the semi-variance at the block size, and reduce the sill accordingly.

Suppose we now decide to regularize this variogram for a larger support, 8 x 8. Following the procedure of §6.1, this can be estimated directly from the empirical variogram, with the appropriate cutoff.

Task 24 : Compute the dispersion variance for 8 x 8 blocks. •

```
> (dv <- mean(variogram(z ~ 1, data = pts.z, cutoff = 8 *
+     sqrt(2), cloud = T)$gamma))

[1] 0.83757
```

Task 25 : Regularize the empirical variogram model. •

This has two steps: (1) reduce the total sill (first taking out the nugget), (2) increase the range by the block size. Recall that for the exponential model the *effective* range is thrice the range *parameter*. So here the block size is 8 x 8, and the corresponding range parameter is $8/3$.

```
> (v.emf.reg <- v.emf)

  model    psill  range
1   Nug 0.047925 0.0000
2   Exp 1.130032 6.3868
```

```
> v.emf.reg[1, "psill"] <- 0
> v.emf.reg[2, "psill"] <- v.emf[2, "psill"] - dv + v.emf[1,
+     "psill"]
> v.emf.reg[2, "range"] <- v.emf.reg[2, "range"] + (8/3)
> print(v.emf.reg)

  model   psill  range
1   Nug 0.00000 0.0000
2   Exp 0.34038 9.0534
```

This is now the expected variogram for further sampling and prediction on 8 x 8 blocks.

Task 26 : Compare this regularized variogram model with the variogram model computed from the 8 x 8 block size, i.e., the 16x16 grid covering the original 128 x 128 grid. •

This was computed above, as part of the series of increasingly-coarse grids:

```
> print(vmf.16)

  model   psill  range
1   Nug 0.00000  0.000
2   Exp 0.58686 15.107
```

Q20 : *How well does the regularized model match the model fitted to the 16 x 16 grid?* *Jump to A20* •

References

[1] R. S. Bivand, E. J. Pebesma, and V. Gómez-Rubio. *Applied Spatial Data Analysis with R*. UseR! Springer, 2008. http://www.asdar-book.org/. 1, 2, 4

[2] J de Gruijter, D J Brus, M F P Bierkens, and M Knotters. *Sampling for Natural Resource Monitoring*. Springer, 2006. 28

[3] C Gotway Crawford and L Young. Change of support: an interdisciplinary challenge. In Philippe Renard, Hélène Demougeot-Renard, and Roland Froidevaux, editors, *Geostatistics for Environmental Applications: Proceedings of the Fifth European Conference on Geostatistics for Environmental Applications*, pages 1–13. Springer, 2005. 1, 2

[4] J. A. Hartigan and M. A. Wong. Algorithm AS136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. 28

[5] E H Isaaks and R M Srivastava. *An introduction to applied geostatistics*. Oxford University Press, New York, 1990. 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. 32

[7] S. Openshaw. *The modifiable areal unit problem*. Concepts and techniques in modern geography,. Geo Books, Norwich, 1983. 2

[8] A. N. Pettitt and A. B. McBratney. Sampling designs for estimating spatial variance components. *Applied Statistics*, 42(1):185–209, 1993. 32

[9] J-M Rendu. *An introduction to geostatistical methods of mineral evaluation*. Geostatistics, 2. South African Institute of Mining and Metallurgy, Johannesburg, 1978. 1, 27

[10] M. Van Meirvenne, T. Meklit, S. Verstraete, M. De Boever, and F. Tack. Could shelling in the first world war have increased copper concentrations in the soil around Ypres? *European Journal of Soil Science*, 59 (2):372–379, 2008. 2

[11] G. M. Vasques, S. Grunwald, N. B. Comerford, and J. O. Sickman. Regional modelling of soil carbon at multiple depths within a subtropical watershed. *Geoderma*, 156(3-4):326–336, 2010. 2

[12] D. Walvoort, D. Brus, and J. de Gruijter. Spatial coverage sampling on various spatial scales. *Pedometron*, 26:20–22, 2009. 28

[13] D. J. J. Walvoort, D. J. Brus, and J. J. de Gruijter. An R package for spatial coverage sampling and random sampling from compact geographical strata by k-means. *Computers & Geosciences*, 36(10):1261–1267, 2010. 28

[14] R. Webster and M. A. Oliver. *Geostatistics for environmental scientists*. Wiley & Sons, Chichester, 2001. 1, 2

[15] 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. 32

Index of R Concepts