

Spatial Coverage Sampling by k -means: the **spcosa**-package*

Dennis Walvoort[†]

Dick Brus[‡]

Jaap de Gruijter[‡]

2009-03-23

Contents

1	Introduction	1
2	Algorithms	2
3	Examples	2
3.1	Spatial coverage sampling without prior points	2
3.2	Spatial coverage sampling with prior points	7
3.3	Random sampling from compact geographical strata	10
3.4	Stratified simple random sampling for composites	13
4	Special cases	16
4.1	Map projections	16
4.2	Simple random sampling	18
4.3	Sampling of non-convex areas	20
A	Class hierarchies	22

1 Introduction

The **spcosa**-package implements algorithms for spatial coverage sampling and for random sampling from compact geographical strata based on the k -means algorithm (de Gruijter et al., 2006). Spatial coverage sampling is known to be an efficient sampling method for model-based mapping (kriging). Random sampling from compact geographical strata is recommended for design-based estimation of spatial means, proportions, *etc.*. In this vignette, the usage of the package will be demonstrated by means of several examples. In addition to spatial coverage sampling also stratified simple random sampling of compact strata will be treated in this vignette.

The **spcosa**-package can be loaded by means of

```
> library(spcosa)
```

Although the implemented optimisation algorithms are deterministic in nature, they use random initial configurations to minimise the risk of ending up in an unfavourable local optimum. In order to be able to reproduce the sampling patterns in a later stage, the built-in pseudo random number generator has to be initialised:

*version: 0.1-2 (2009-03-23)

[†]Alterra - Centre for Water & Climate, Wageningen University and Research Centre, The Netherlands

[‡]Alterra - Soil Science Centre, Wageningen University and Research Centre, The Netherlands

```
> set.seed(314)
```

The `spcosa`-package depends on the `sp`-package (Pebesma and Bivand, 2005) for storing spatial information, and the `ggplot2`-package (Wickham, 2008) for visualisation. A basic knowledge of the `sp`-package is highly recommended. See the [sp-package vignette](#) for more information. Knowledge of the `ggplot2`-package is only needed for fine-tuning `spcosa` graphics. Consult the superb [ggplot2-website](#) for details and illustrative examples.

2 Algorithms

The basic idea is to distribute sampling points evenly over the study area by selecting these points in compact spatial strata. Compact strata can be obtained by k -means clustering of the cells making up a fine grid representing the study area of interest. Two k -means algorithms have been implemented in the `spcosa`-package: a transfer algorithm and a swapping algorithm. The transfer algorithm obtains compact clusters (strata) by transferring cells from one cluster to the other, whereas the swapping algorithm achieves this by swapping cells between clusters. The first algorithm results in compact clusters, whereas the second algorithm results in compact clusters of equal size. For reasons of efficiency, both algorithms have been implemented in the [Java](#) language and communicate with R (R Development Core Team, 2008) by means of the `rJava`-package (Urbanek, 2008).

3 Examples

In this section, the `spcosa`-package will be demonstrated by means of several examples.

3.1 Spatial coverage sampling without prior points

In the first example, spatial coverage sampling will be applied to map the clay content in the study area by means of ordinary point kriging.

First, a vector or raster representation of the study area has to be loaded. As an example, the ASCII-grid `demoGrid.asc` residing in the `maps` directory of the `spcosa`-package will be read by means of the `readGDAL`-function of the `rgdal`-package (Keitt et al., 2008).

```
> filename <- system.file("maps", "demoGrid.asc", package = "spcosa")
> myAsciiGrid <- readGDAL(fname = filename, silent = TRUE)
```

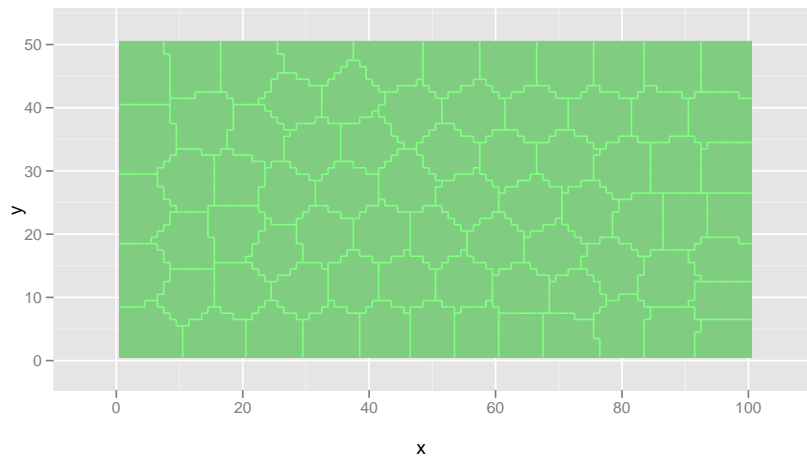
To obtain a uniform distribution of sampling points over the study area, the sampling points will be selected at the centroids of compact spatial strata. Compact strata can be constructed by invoking the `stratify` method:

```
> myStratification <- stratify(myAsciiGrid, nStrata = 75, nTry = 10)
> myStratification
```

```
Object of class "CompactStratification"
number of strata: 75
```

In this example, the study area has been partitioned into 75 compact strata. The resulting stratification can be plotted by means of:

```
> print(plot(myStratification))
```



Note that throughout this vignette, `plot` statements are encapsulated in `print` statements. The reason for this is explained in the [online Sweave FAQ](#). The `print` statements can be omitted during an ordinary interactive R-session.

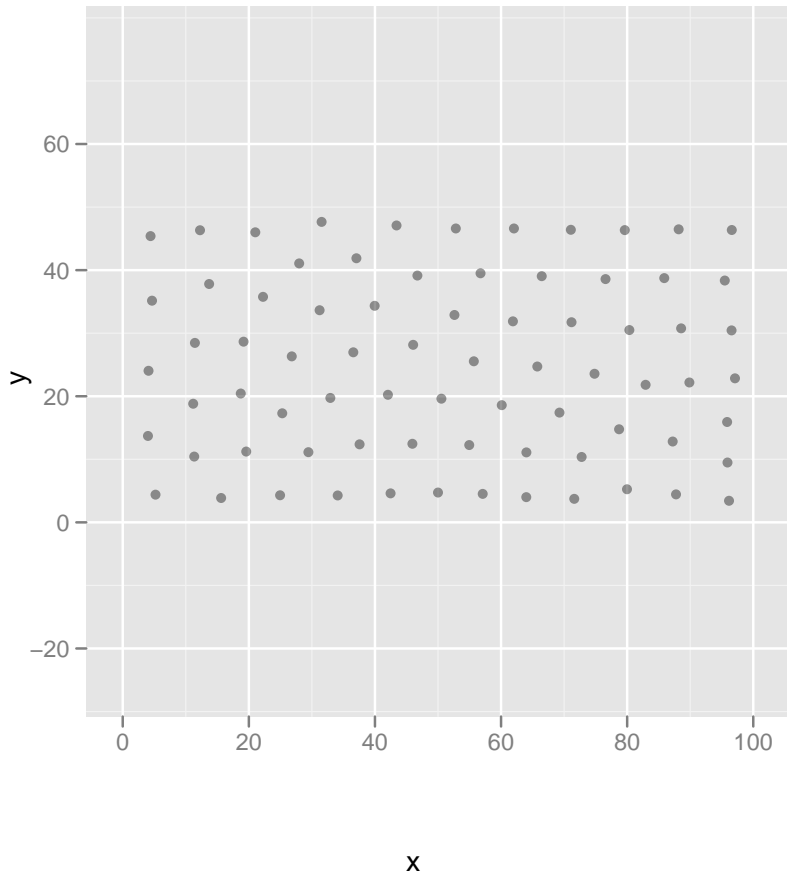
The `spsample`-method, an overloaded method from the `sp`-package, can be used to select the centroid of each stratum:

```
> mySamplingPattern <- spsample(myStratification)
> mySamplingPattern
```

```
Object of class "SamplingPatternCentroids"
sample size: 75
```

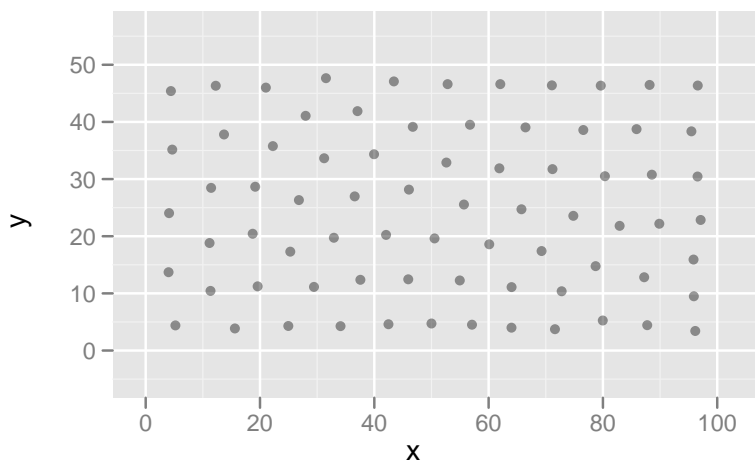
The `plot`-method can be used to visualise the resulting sampling pattern:

```
> print(plot(mySamplingPattern))
```



Note that the `plot`-method creates maps with equally scaled axes to prevent distortion. Unfortunately, this may occasionally lead to wide margins, as in the example above. This can be corrected by specifying the proper aspect ratio:

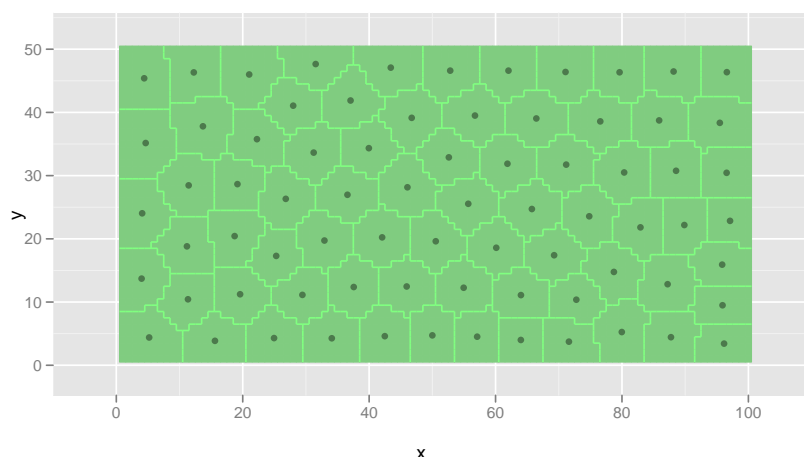
```
> print(plot(mySamplingPattern) + coord_equal(ratio = 3/5))
```



Indeed, each plot can be modified by adding `ggplot2`-functions, like `coord_equal`, to the `plot`-method.

The sampling pattern can also be plotted on top of the stratification:

```
> print(plot(myStratification, mySamplingPattern))
```



Sampling points can be extracted by means of a simple type cast to class `data.frame`:

```
> mySamplingPoints <- as(mySamplingPattern, "data.frame")
```

Next, field work will be done to acquire data at these locations. In this vignette, however, we simulated data on clay and soil organic matter contents by calling the function `doFieldWork`. The `doFieldWork`-function is not part of the `spcosa`-package, it only ‘lives’ in this vignette.

```
> myData <- doFieldWork(mySamplingPoints)
> str(myData)
```

```
'data.frame':      75 obs. of  2 variables:
 $ clay: num  8 7.6 13.5 11.7 10.7 11.3 15 10.5 8.1 12.5 ...
 $ SOM : num  4.4 4 6.9 6 5.4 6.1 8.1 5.2 4.2 6.8 ...
```

Spatial coverage sampling is a purposive sampling method. Therefore, model-based inference will be applied to predict clay contents. Model-based inference is not implemented in the `spcosa`-package, because other add-on packages already provide excellent support for it. See for instance, `gstat` (Pebesma, 2004), `spatial` (Venables and Ripley, 2002), and `geoR` (Ribeiro Jr and Diggle, 2001).

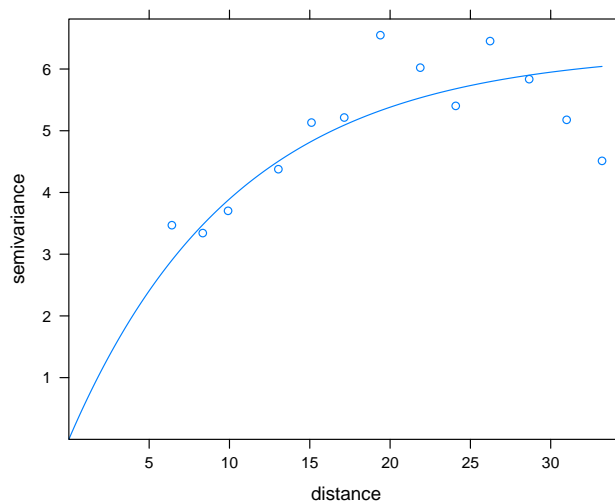
As an example, we will use the `gstat` package for predicting clay contents by means of ordinary point kriging. First, the observations and sampling pattern will be combined in a single object of class `SpatialPointsDataFrame`:

```
> spData <- data.frame(mySamplingPoints, myData)
> coordinates(spData) <- ~x * y
```

Next, the semivariogram model for clay has to be estimated:

```
> sampleVariogram <- variogram(clay ~ 1, spData)
> variogramModel <- fit.variogram(sampleVariogram, model = vgm(psill = 1,
+   model = "Exp", range = 5, nugget = 0))
```

```
> print(plot(sampleVariogram, model = variogramModel))
```



Finally, ordinary point kriging predictions can be obtained by means of:

```
> g <- gstat(id = "clay", formula = clay ~ 1, data = spData, model = variogramModel)
> yHat <- predict(g, newdata = myAsciiGrid)
```

[using ordinary kriging]

```
> summary(yHat)
```

Object of class SpatialGridDataFrame

Coordinates:

min max

x 0.5 100.5

y 0.5 50.5

Is projected: NA

proj4string : [NA]

Number of points: 2

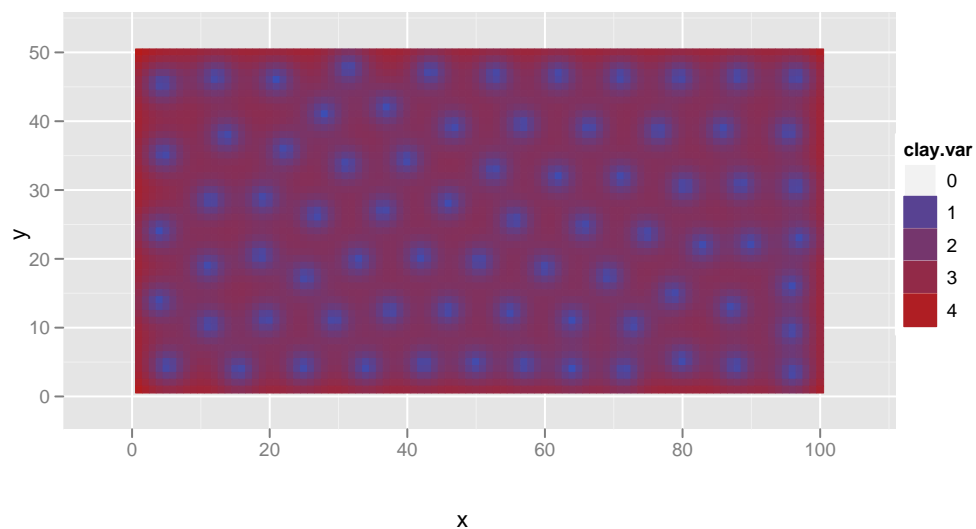
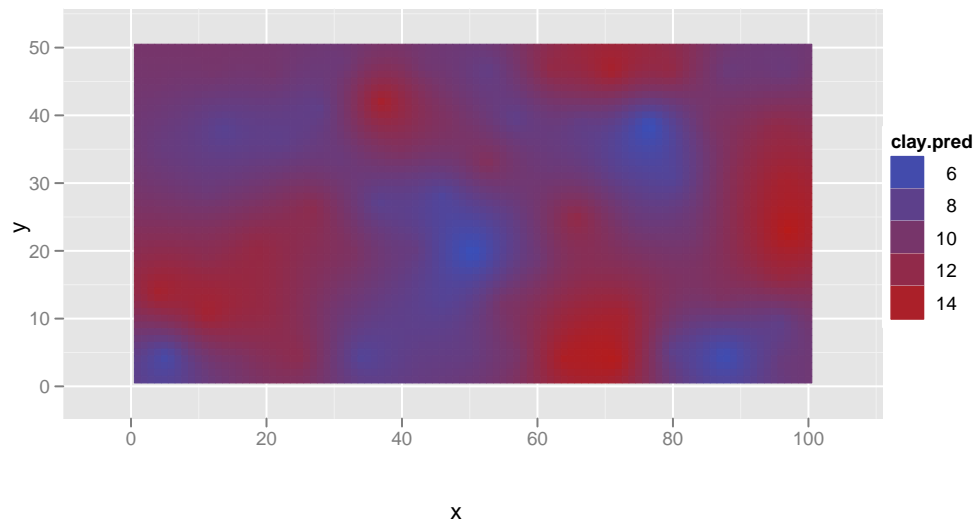
Grid attributes:

	cellcentre.offset	cellsize	cells.dim
x	1	1	100
y	1	1	50

Data attributes:

	clay.pred	clay.var
Min.	: 5.305	Min. :2.519e-32
1st Qu.:	8.810	1st Qu.:1.907e+00
Median :	9.805	Median :2.260e+00
Mean :	9.995	Mean :2.154e+00
3rd Qu.:	11.181	3rd Qu.:2.446e+00
Max.	:14.931	Max. :4.261e+00

The resulting predictions and variances of the prediction error are given in the figures below:



3.2 Spatial coverage sampling with prior points

Sometimes, samples from previous sampling campaigns are available. In these situations, spatial infill sampling may be performed. This type of spatial coverage sampling aims to distribute new sampling points evenly over the study area, while taking the locations of existing sampling points into account. Suppose a `data.frame` is available containing the coordinates of 50 existing sampling points:

```
> str(priorPoints)

'data.frame':      50 obs. of  2 variables:
 $ x: num  9.5 53.1 18.1 56 64.6 90.3 83.6 76.4 1.4 24.1 ...
 $ y: num  14.6 9.4 38.5 37.6 1 1.3 9.2 40.3 9 27.4 ...
```

Twenty-five *new* points can be assigned to sparsely sampled regions by means of:

```

> coordinates(priorPoints) <- ~x * y
> myStratification <- stratify(myAsciiGrid, priorPoints = priorPoints,
+   nStrata = 75, nTry = 100)
> mySamplingPattern <- spsample(myStratification)

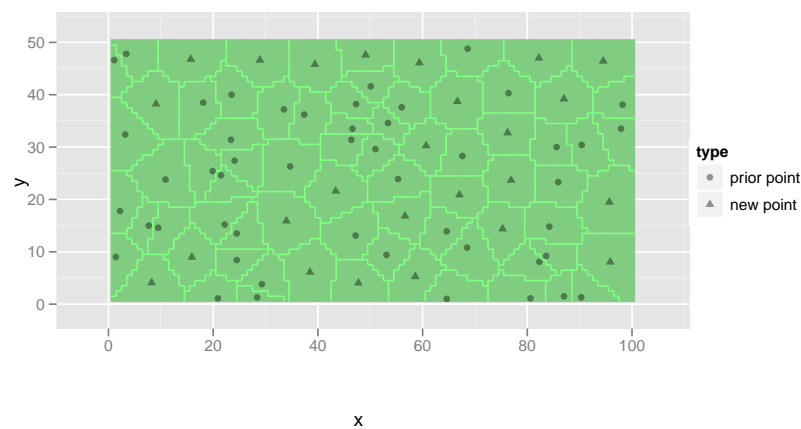
```

Note that the total number of strata, and therefore the total number of points, equals $50 + 25 = 75$. In addition, also note that the `nTry` argument has been set to 100. The algorithm will now use 100 random starting configurations and keeps the best solution to minimise the risk of ending up in an unfavourable local optimum.

```

> print(plot(myStratification, mySamplingPattern))

```



Note that prior points and new points are represented by different symbols.

A map of clay contents can be obtained by means of ordinary point kriging. First, some field and laboratory work has to be done to obtain clay contents for the 25 new points:

```

> mySamplingPoints <- as(mySamplingPattern, "data.frame")
> myData <- doFieldWork(mySamplingPoints)
> spData <- data.frame(mySamplingPoints, myData["clay"])
> coordinates(spData) <- ~x * y
> summary(spData)

```

Object of class `SpatialPointsDataFrame`

Coordinates:

```

  min max
x 1.1 98.2
y 1.0 48.8

```

Is projected: NA

proj4string : [NA]

Number of points: 75

Data attributes:

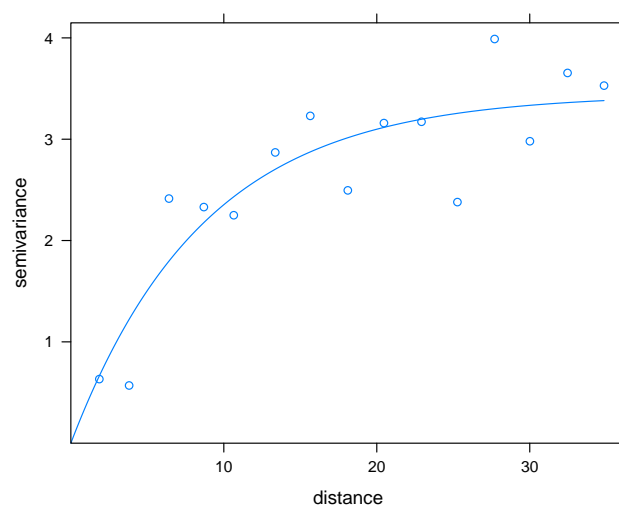
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
5.00	8.85	10.00	10.02	11.40	15.00

Again, we will use `gstat` for model-based inference:


```

> sampleVariogram <- variogram(clay ~ 1, spData)
> variogramModel <- fit.variogram(sampleVariogram, model = vgm(psill = 1,
+   model = "Exp", range = 1, nugget = 0))
> print(plot(sampleVariogram, model = variogramModel))

```



```

> g <- gstat(id = "clay", formula = clay ~ 1, data = spData, model = variogramModel)
> yHat <- predict(g, newdata = myAsciiGrid)

```

[using ordinary kriging]

```
> summary(yHat)
```

Object of class SpatialGridDataFrame

Coordinates:

min max

x 0.5 100.5

y 0.5 50.5

Is projected: NA

proj4string : [NA]

Number of points: 2

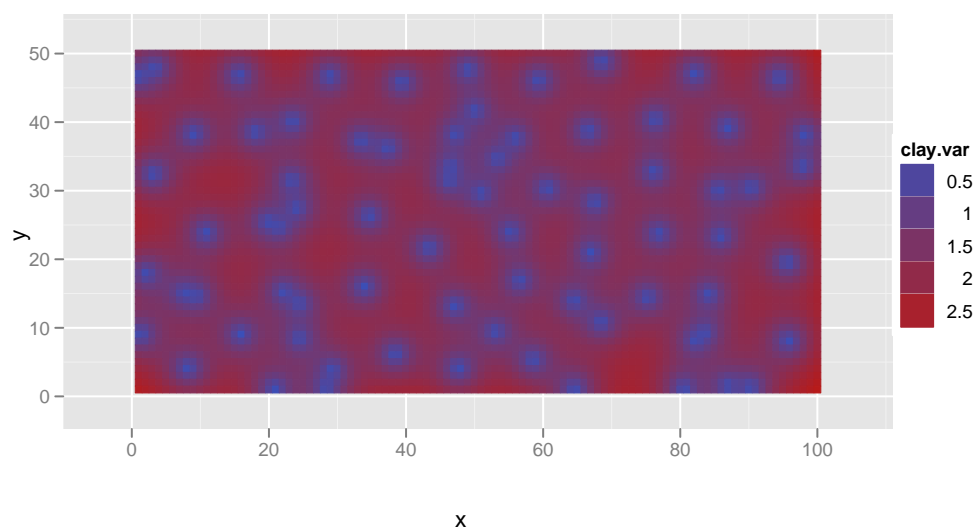
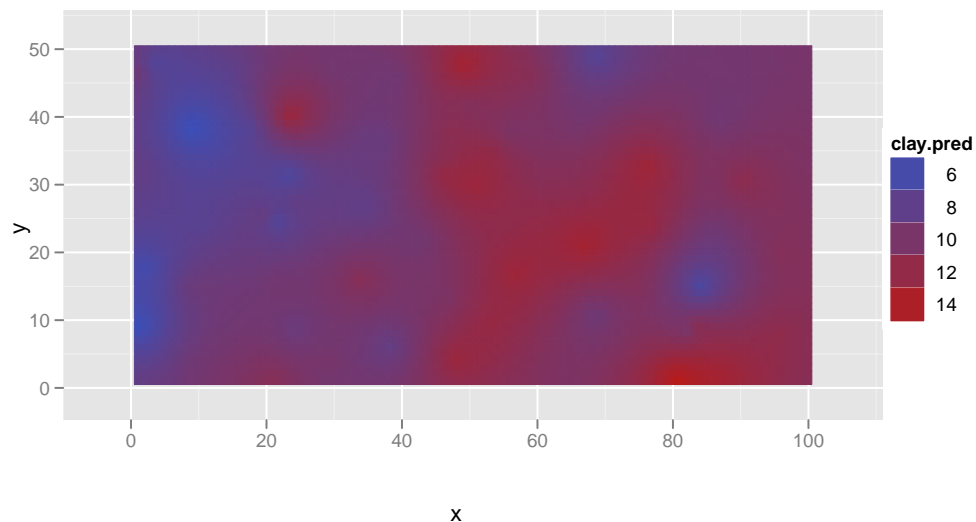
Grid attributes:

	cellcentre.offset	cellsize	cells.dim
x	1	1	100
y	1	1	50

Data attributes:

	clay.pred	clay.var
Min.	: 5.124	Min. :0.05476
1st Qu.:	9.030	1st Qu.:1.24923
Median :	9.969	Median :1.56104
Mean :	9.928	Mean :1.49111
3rd Qu.:	11.028	3rd Qu.:1.76897
Max.	:14.802	Max. :2.83482

The resulting predictions and variances of the prediction error are given in the figures below:



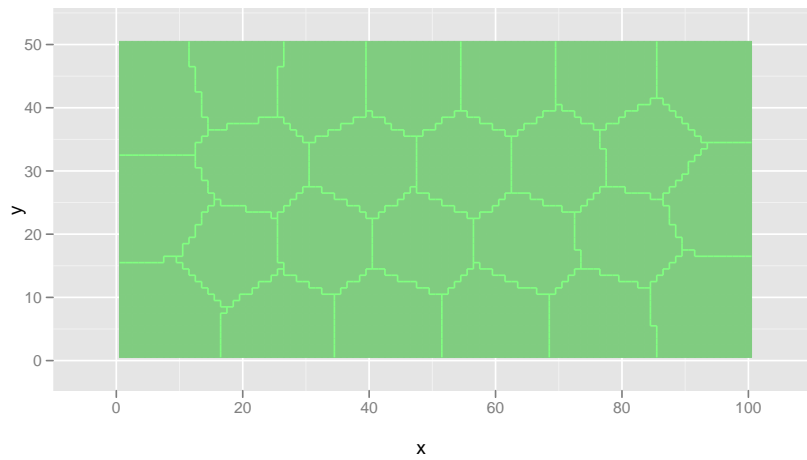
3.3 Random sampling from compact geographical strata

In this section the global mean clay and organic matter contents of the study area will be estimated by means of stratified simple random sampling. Again, we will use `myAsciiGrid` as a representation of the study area. The study area will be partitioned into 25 compact strata:

```
> myStratification <- stratify(myAsciiGrid, nStrata = 25, nTry = 10)
> myStratification

Object of class "CompactStratification"
number of strata: 25

> print(plot(myStratification))
```

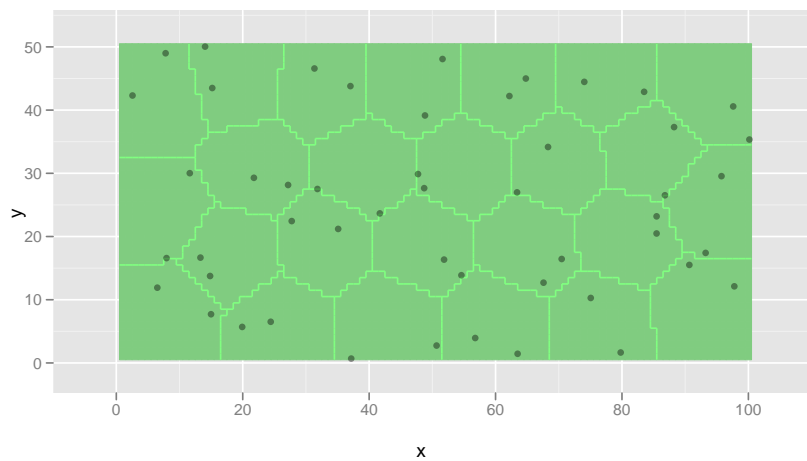


The `spsample`-method can be used to randomly sample two sampling units per stratum:

```
> mySamplingPattern <- spsample(myStratification, n = 2)
> mySamplingPattern
```

Object of class "SamplingPatternRandomSamplingUnits"
sample size: 50

```
> print(plot(myStratification, mySamplingPattern))
```



Sampling points can be extracted by means of a type cast to class `data.frame`:

```
> mySamplingPoints <- as(mySamplingPattern, "data.frame")
> mySamplingPoints[1:5, ]
```

	x	y
1	63.48984	1.449016
2	56.78113	3.942664
3	48.70803	27.649526
4	47.73079	29.879614
5	14.98596	7.704725

Next, some field work has to be done:

```
> myData <- doFieldWork(mySamplingPoints)
> str(myData)

'data.frame':      50 obs. of  2 variables:
 $ clay: num  10.4 9.5 14.7 14.1 10.3 8.4 10.5 10.9 11.5 8.9 ...
 $ SOM : num  5.7 4.7 7.1 6.7 5.5 3.8 4.9 5.2 5.6 4.3 ...
```

The spatial mean clay and soil organic matter contents can be estimated by (de Gruijter et al., 2006):

```
> estimate("spatial mean", myStratification, mySamplingPattern,
+         myData)

      clay      SOM
10.20568  5.05320
```

In estimating the spatial mean, differences in surface area of the strata are taken into account. Note, that the spatial mean is estimated for all columns in `myData`. The standard error can be estimated in a similar way:

```
> estimate("standard error", myStratification, mySamplingPattern,
+         myData)

      clay      SOM
0.2050475 0.1148747
```

The spatial cumulative distribution function (SCDF) (see de Gruijter et al., 2006) can be estimated by means of

```
> mySCDF <- estimate("scdf", myStratification, mySamplingPattern,
+         myData)
```

The SCDFs are returned as a list of matrices, *i.e.* one matrix for each property:

```
> lapply(X = mySCDF, FUN = head, n = 4)

$clay
      value cumFreq
[1,]   5.0  0.0000
[2,]   6.6  0.0173
[3,]   7.5  0.0385
[4,]   7.8  0.0587

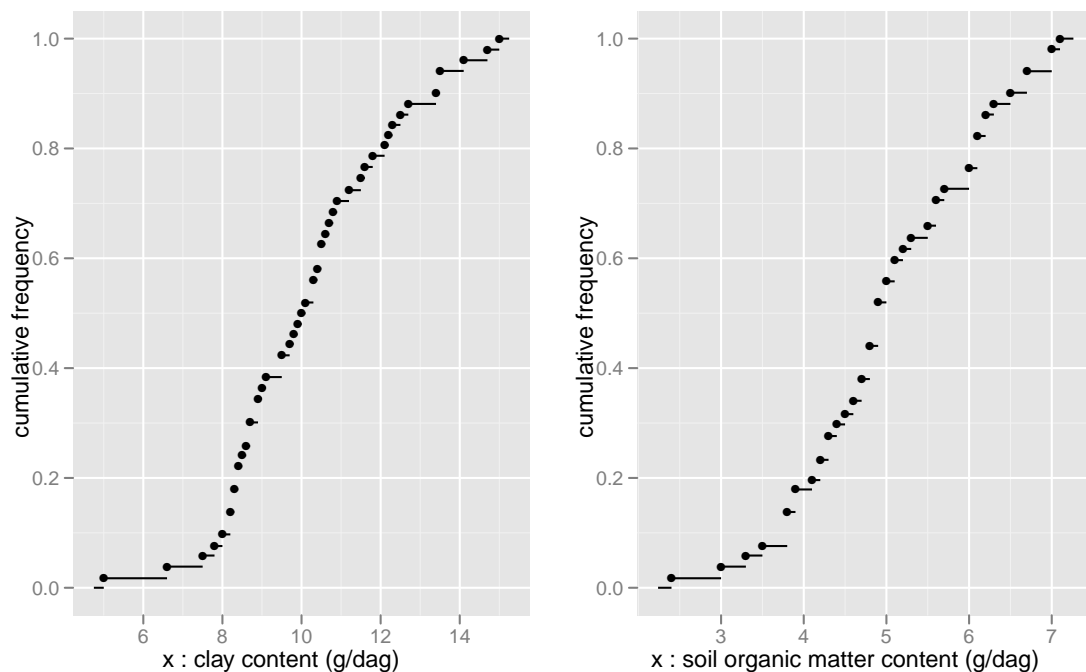
$SOM
      value cumFreq
[1,]   2.4  0.0000
[2,]   3.0  0.0173
[3,]   3.3  0.0385
[4,]   3.5  0.0587

> lapply(X = mySCDF, FUN = tail, n = 4)
```

```
$clay
      value cumFreq
[38,]  13.5  0.9008
[39,]  14.1  0.9412
[40,]  14.7  0.9606
[41,]  15.0  0.9800
```

```
$SOM
      value cumFreq
[27,]   6.5  0.8811
[28,]   6.7  0.9015
[29,]   7.0  0.9406
[30,]   7.1  0.9806
```

The SCDFs for clay and SOM are visualised below.



3.4 Stratified simple random sampling for composites

In this example the aim is to estimate the global mean clay and organic matter contents of a field. To reduce laboratory costs, the soil aliquots collected at the sampling locations will be bulked into composite samples. The study area is a field near the village of Farmsum, in the North-East of the Netherlands. An ESRI shape file of this field is available in the `maps` directory of the `spcosa`-package. It can be loaded by means of `readOGR`, a function in the `rgdal`-package (Keitt et al., 2008):

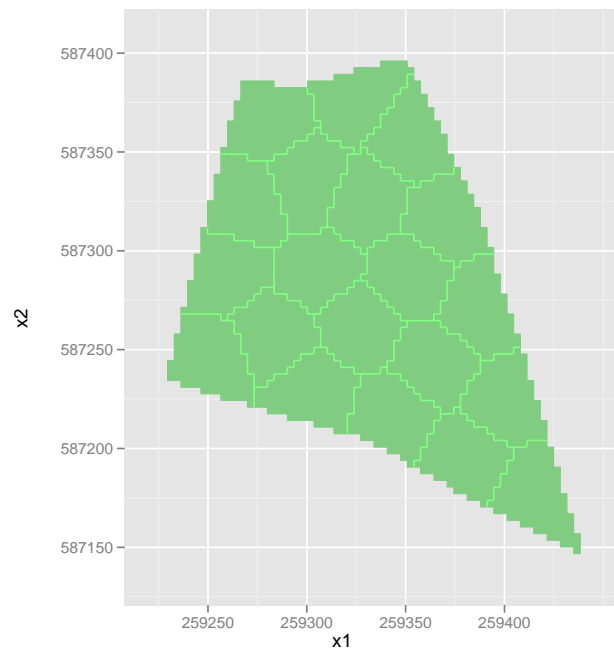
```
> directory <- system.file("maps", package = "spcosa")
> shpFarmsum <- readOGR(dsn = directory, layer = "farmsum", verbose = FALSE)
```

First, the field will be stratified into, say, 20 compact strata of equal size. Strata of equal size are desirable to simplify fieldwork, *i.e.* equal masses (or volumes) of soil can be collected at the sampling locations (Brus et al., 1999; de Gruijter et al., 2006).

```
> myStratification <- stratify(shpFarmsum, nStrata = 20, equalArea = TRUE,
+   nTry = 10)
> myStratification
```

```
Object of class "CompactStratificationEqualArea"
number of strata: 20
```

```
> print(plot(myStratification))
```

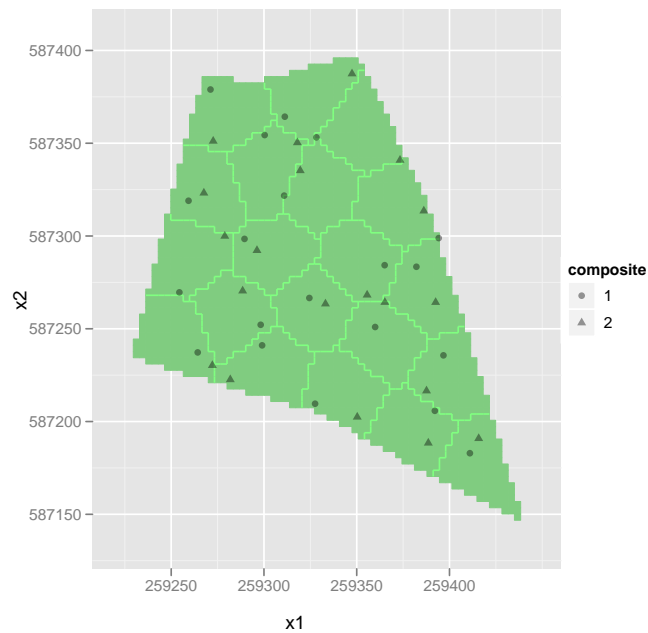


Next, two sampling units will be selected at random in each stratum. At least two sampling units per stratum are required to estimate the sampling variance of the estimated mean.

```
> mySamplingPattern <- spsample(myStratification, n = 2, type = "composite")
> mySamplingPattern
```

```
Object of class "SamplingPatternRandomComposite"
sample size: 2
```

```
> print(plot(myStratification, mySamplingPattern))
```



Sampling points can be extracted means of:

```
> mySamplingPoints <- as(mySamplingPattern, "data.frame")
> mySamplingPoints[1:5, ]
```

	composite	x1	x2
1	1	259365.0	587284.3
2	2	259355.6	587268.2
3	1	259324.5	587266.6
4	2	259333.1	587263.4
5	1	259289.5	587298.4

Note that an extra column has been added specifying the sampling units to be bulked into each composite. A composite sample is formed by bulking one aliquot (sampling unit) per stratum. Field work now results in a composite sample of size two:

```
> myData <- doFieldWork(mySamplingPoints, composite = TRUE, n = 2)
> myData
```

	clay	SOM
1	9.2	4.7
2	8.9	4.5

The spatial mean and its standard error can be estimated by means of:

```
> estimate("spatial mean", myStratification, mySamplingPattern,
+         myData)

clay  SOM
9.05  4.60

> estimate("standard error", myStratification, mySamplingPattern,
+         myData)
```

```

      clay      SOM
0.0225 0.0100

```

If we do not want to bulk soil aliquots, the same stratification can be used to select a sample of 20×2 sampling locations:

```

> mySamplingPattern <- spsample(myStratification, n = 2)
> mySamplingPoints <- as(mySamplingPattern, "data.frame")
> mySamplingPoints[1:5, ]

```

```

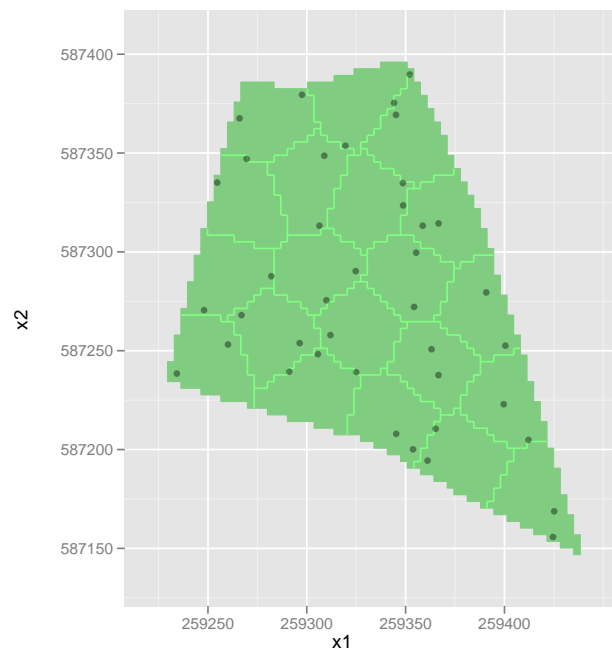
      x1      x2
1 259354.3 587272.1
2 259355.3 587299.6
3 259325.0 587239.1
4 259312.0 587257.9
5 259324.7 587290.3

```

```

> print(plot(myStratification, mySamplingPattern))

```



4 Special cases

4.1 Map projections

In the examples above, the maps didn't have projection attributes. At field scale, projection attributes are not really necessary. However, at continental and global scale, for example, projection attributes can't be ignored. The `spcosa`-package is capable of handling projection attributes of class `CRS`. More information on projections is available at the ['PROJ.4' project website](#) and in the `rgdal`-package documentation ([Keitt et al., 2008](#)).

To illustrate the effect of stratification on smaller spatial scales, consider two grids covering the entire Northern Hemisphere.


```

> grd1 <- expand.grid(longitude = seq(from = -178, to = 180, by = 2),
+   latitude = seq(from = 2, to = 88, by = 2))
> coordinates(grd1) <- ~longitude * latitude
> gridded(grd1) <- TRUE
> grd2 <- grd1
> proj4string(grd2) <- CRS("+proj=longlat +ellps=WGS84")

```

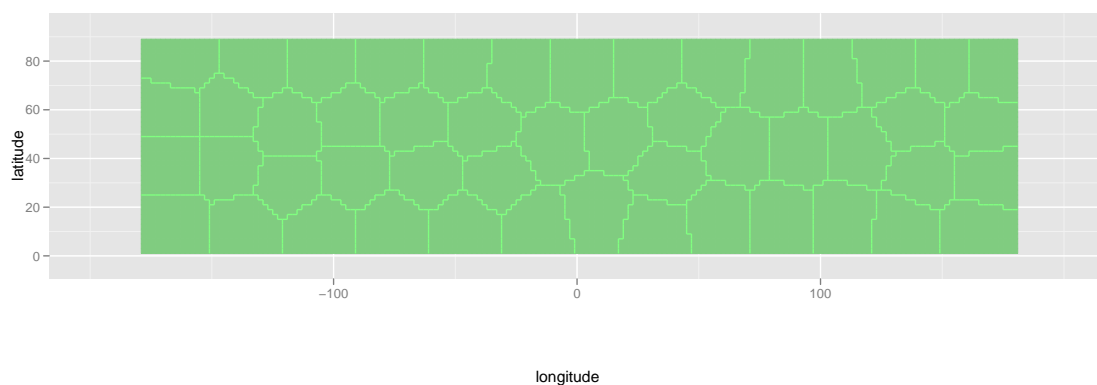
Note that `grd1` is identical to `grd2`, except that `grd2` has projection attributes. If no projection attributes are available, the algorithms in the `spcosa`-package use squared Euclidean distances in the *k*-means algorithms. However, if projection attributes have been set, the coordinates will be transformed to lat/long format (latitude/longitude) and squared great circle distances will be used instead.

Both grids will be partitioned into 50 compact strata:

```

> myStratification1 <- stratify(grd1, nStrata = 50)
> myStratification2 <- stratify(grd2, nStrata = 50)
> print(plot(myStratification1) + coord_equal(ratio = 90/360))

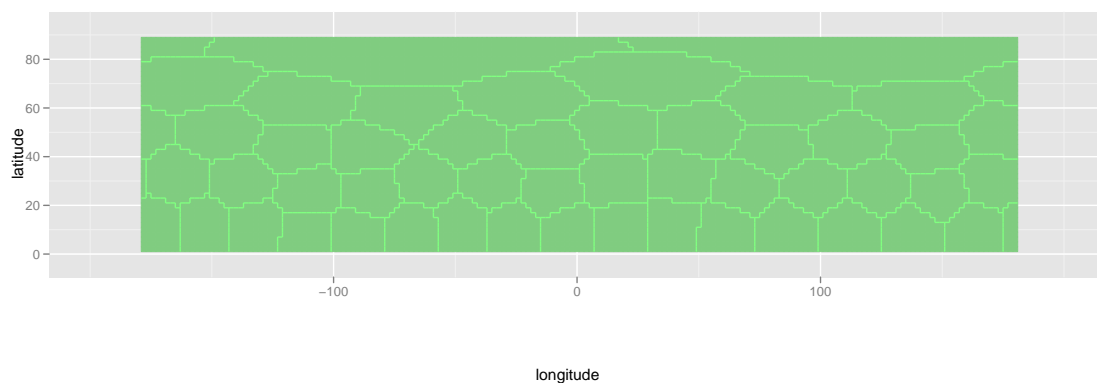
```



```

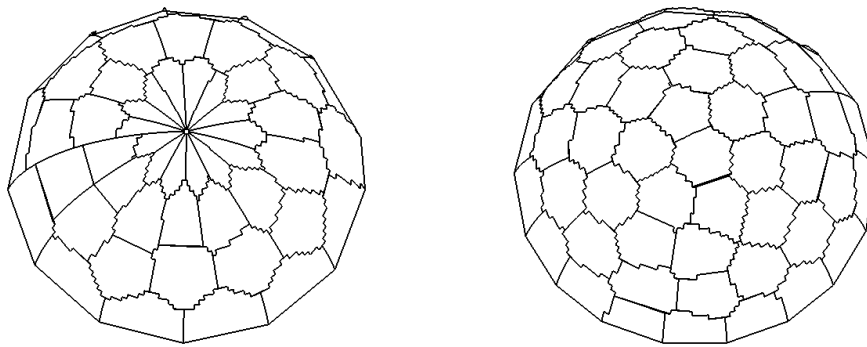
> print(plot(myStratification2) + coord_equal(ratio = 90/360))

```



Note that `grd1` seems to have more compact strata near the geographic north pole than `grd2`. However, the contrary is true. This becomes evident when both stratifications are projected on a

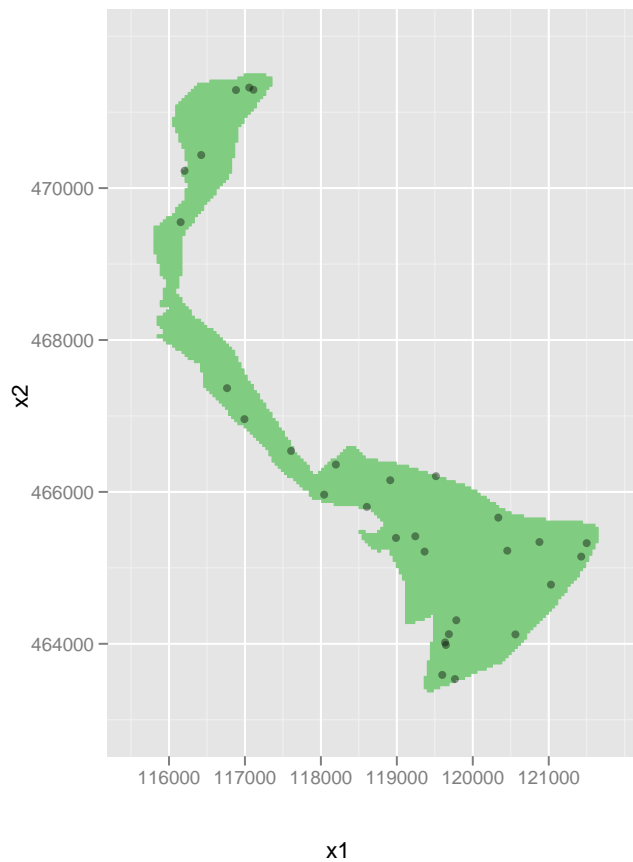
sphere (see below). The left figure corresponds to **grd1** and the right figure to **grd2**. The strata of **grd2** are clearly more compact than those of **grd1**. In addition, **grd1** suffers from pronounced edge effects near the poles and at 180° longitude. The strata are discontinuous at this meridian, *i.e.*, two points on opposite sides of the meridian are treated as very distant when squared Euclidean distances are used. The strata of **grd2**, on the other hand, have been optimised on a sphere by using squared great circle distances, and don't suffer from edge effects at all, *i.e.*, the distance between two points on opposite sides of this meridian is very small.



4.2 Simple random sampling

Although this package is about sampling from compact strata, it can also be used for simple random sampling, by setting `nStrata = 1`:

```
> shpMijdrecht <- readOGR(dsn = system.file("maps", package = "spcosa"),
+   layer = "mijdrecht", verbose = FALSE)
> myStratification <- stratify(shpMijdrecht, nStrata = 1, nGridCells = 5000)
> mySamplingPattern <- spsample(myStratification, n = 30)
> print(plot(myStratification, mySamplingPattern))
```



```
> mySamplingPoints <- as(mySamplingPattern, "data.frame")
> myData <- doFieldWork(mySamplingPoints)
> str(myData)
```

```
'data.frame':      30 obs. of  2 variables:
 $ clay: num  6.3 13.3 8.2 6.4 6.2 5.9 5 8.1 5.2 15 ...
 $ SOM : num  3.3 6.9 4 3 2.9 2.8 2.8 4.2 3.2 7.5 ...
```

The spatial mean and its standard error can be estimated by:

```
> estimate("spatial mean", myStratification, mySamplingPattern,
+         myData)
```

```
      clay      SOM
7.876667 3.930000
```

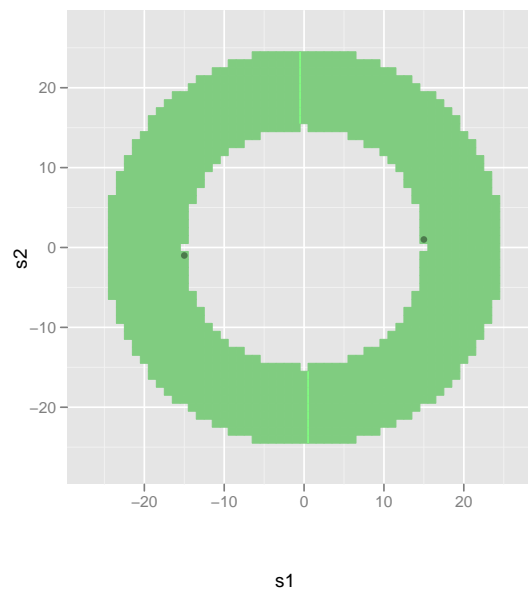
```
> estimate("standard error", myStratification, mySamplingPattern,
+         myData)
```

```
      clay      SOM
0.6172554 0.3246979
```

4.3 Sampling of non-convex areas

In case of spatial coverage sampling (*e.g.*, Sections 3.1 and 3.2), sampling the centroid of each cluster may become problematic in case of non-convex areas. A centroid may be situated well outside the area of interest. If this happens, the sampling point will be relocated to the nearest grid cell that is part of the target universe. This pragmatic solution usually gives reasonable results. However, in some extreme situations the solution may be less desirable. As an example, consider the ‘doughnut’-shaped field below.

```
> myStratification <- stratify(doughnut, nStrata = 2, nTry = 100)
> mySamplingPattern <- spsample(myStratification)
> print(plot(myStratification, mySamplingPattern))
```



Note that this problem does not arise in random sampling from compact geographical strata (*e.g.*, Sections 3.3 and 3.4).

References

- D. J. Brus, L. E. E. M. Spätjens, and J. J. de Gruijter. A sampling scheme for estimating the mean extractable phosphorus concentration of fields for environmental regulation. *Geoderma*, 89:129–148, 1999.
- J. de Gruijter, D. Brus, M. Bierkens, and M. Knotters. *Sampling for Natural Resource Monitoring*. Springer, Berlin, 2006.
- T. H. Keitt, R. Bivand, E. Pebesma, and B. Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2008. URL <http://www.gdal.org>, <http://rgdal.sourceforge.net/>, <http://sourceforge.net/projects/rgdal/>. R package version 0.5-33.
- E. J. Pebesma. Multivariable geostatistics in s: the gstat package. *Computers & Geosciences*, 30: 683–691, 2004.

- E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL <http://CRAN.R-project.org/doc/Rnews/>.
- R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL <http://www.R-project.org>. ISBN 3-900051-07-0.
- P. J. Ribeiro Jr and P. J. Diggle. geoR: a package for geostatistical analysis. *R-NEWS*, 1(2):14–18, June 2001. URL <http://CRAN.R-project.org/doc/Rnews/>. ISSN 1609-3631.
- S. Urbanek. *rJava: Low-level R to Java interface*, 2008. URL <http://www.rforge.net/rJava/>. R package version 0.6-0.
- W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL <http://www.stats.ox.ac.uk/pub/MASS4>. ISBN 0-387-95457-0.
- H. Wickham. *ggplot2: An implementation of the Grammar of Graphics*, 2008. URL <http://had.co.nz/ggplot2/>. R package version 0.8.1.

A Class hierarchies

The `spcosa`-package uses **S4-type classes and methods**. The class hierarchies are visualised below.

