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

Dennis Walvoort† Dick Brus† Jaap de Gruijter†

2012-09-11

## Contents



## 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; Walvoort et al., 2010). Spatial coverage sampling is known to be an efficient sampling method

*version: 0.3-1 (2012-09-11)

†Alterra - Soil Science Centre, Wageningen University and Research Centre, The Netherlands

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 `spocsa`-package can be loaded by means of

```
library(spcosa)


## Loading required package:  rJava

## Loading required package:  methods

## Loading required package:  ggplot2

## Note:  'spcosa' requires Java (>= 6.0), which is available at www.java.com
```

Although the implemented optimisation algorithms are deterministic in nature, they use a user-specified number (`nTry`, see examples below) of random initial configurations to reduce 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 pseudo random number generator of R has to be initialised first:

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

The `spcosa`-package depends on the `sp`-package (Pebesma and Bivand, 2005) for storing spatial information, and the `ggplot2`-package (Wickham, 2009) 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 swopping algorithm (Walvoort et al., 2010). The transfer algorithm obtains compact clusters (strata) by transferring cells from one cluster to the other, whereas the swopping algorithm achieves this by swopping 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, 2012) by means of the `rJava`-package (Urbanek, 2011).

# 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 a 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., 2012).

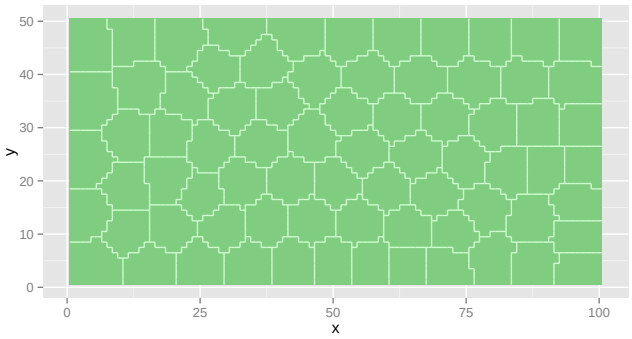
```
library(rgdal)


## Loading required package:  sp

## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime:  GDAL 1.9.1, released 2012/05/15
## Path to GDAL shared files:  d:/R/lib/rgdal/gdal
## Loaded PROJ.4 runtime:  Rel.  4.7.1, 23 September 2009, [PJ_VERSION: 470]
## Path to PROJ.4 shared files:  d:/R/lib/rgdal/proj


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:

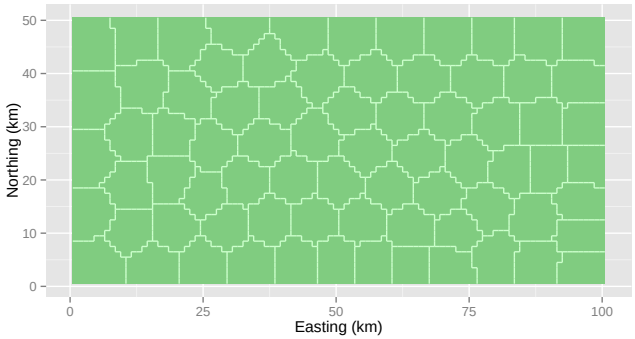
```
plot(myStratification)
```




Each plot can be modified by adding `ggplot2`-functions to the `plot`-method:

```
plot(myStratification) +
    scale_x_continuous(name = "Easting (km)") +
    scale_y_continuous(name = "Northing (km)")
```

```
## Scale for 'x' is already present.  Adding another scale for 'x', which will replace the existing
scale.

## Scale for 'y' is already present.  Adding another scale for 'y', which will replace the existing
scale.
```

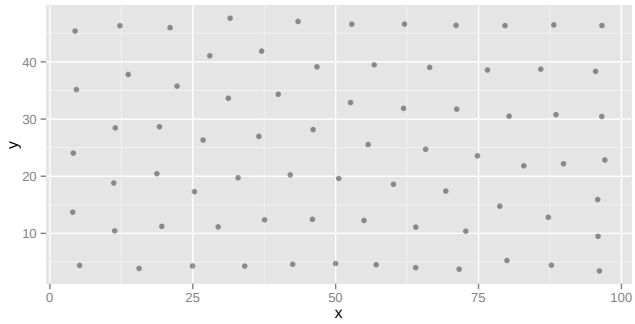



The `spsample`-method, an overloaded method from the `sp`-package, can be used to select the centroid of each stratum:

```
mySamplingPattern <- spsample(myStratification)
mySamplingPattern


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

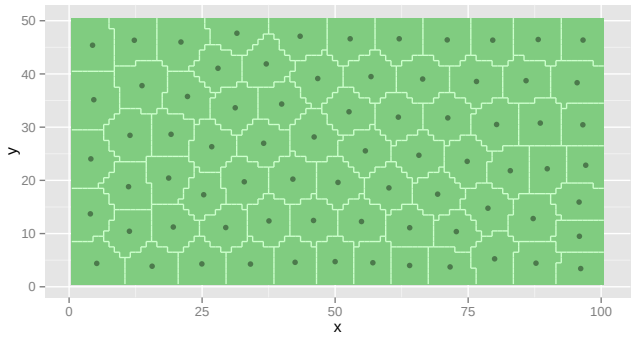
The `plot`-method can be used to visualise the resulting sampling pattern:

```
plot(mySamplingPattern)
```




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

```
plot(myStratification, mySamplingPattern)
```




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

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

```
##        x      y
## 1 79.972  5.254
## 2 13.704 37.802
## 3 11.339 10.435
## 4 26.809 26.324
## 5  4.091 24.039
## 6 25.298 17.298
```

Next, field work will be done to acquire data at these locations. In this vignette, however, we will simulate 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)
```

```
## Loading required package:  gstat

## Loading required package:  spacetime

## Loading required package:  zoo

##
## Attaching package:  'zoo'

## The following object(s) are masked from 'package:base':
##
## as.Date, as.Date.numeric

## Loading required package:  xts
```

```
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).

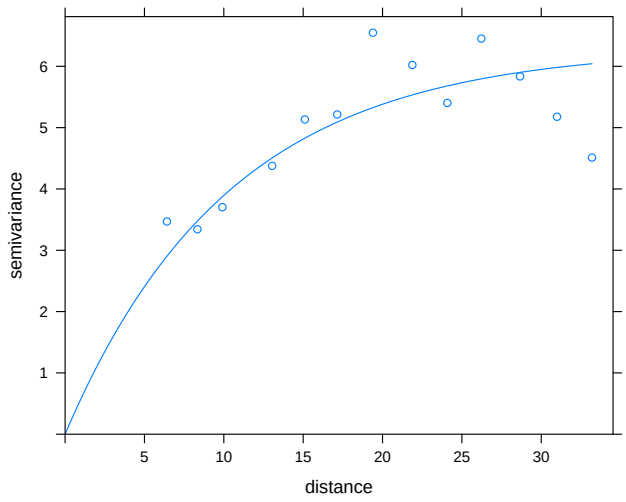
As an example, we will use the `gstat` package for predicting clay contents by means of ordinary point kriging. First, the observations and sampling pattern will be combined in a single object of class `SpatialPointsDataFrame`:

```
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))
```

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

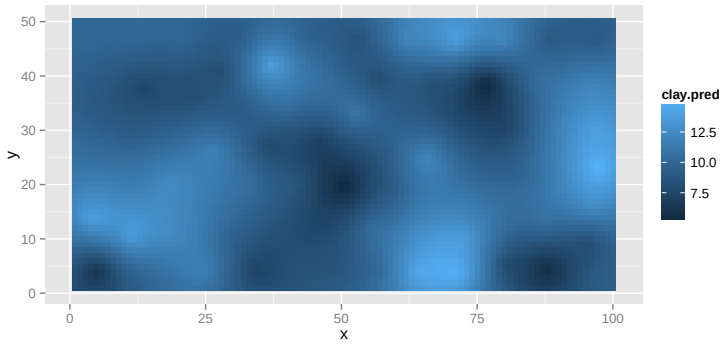



Finally, ordinary point kriging predictions can be obtained by means of:

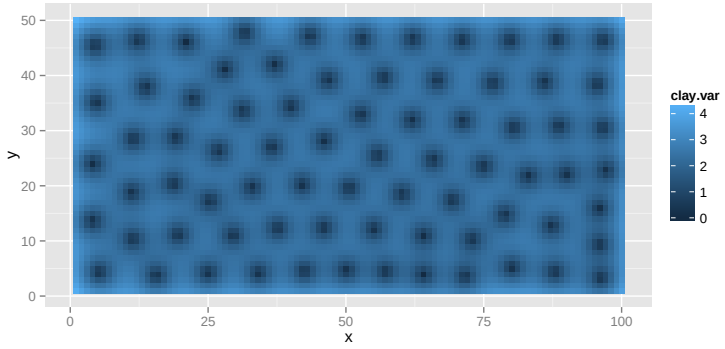
```
g <- gstat(id = "clay", formula = clay ~ 1, data = spData, model = variogramModel)
yHat <- predict(g, newdata = myAsciiGrid)


## [using ordinary kriging]
```

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

```
ggplot(data = as(yHat, "data.frame")) +
    geom_raster(mapping = aes(x = x, y = y, fill = clay.pred, colour = clay.pred)) +
    coord_equal()
```




```
ggplot(data = as(yHat, "data.frame")) +
    geom_raster(mapping = aes(x = x, y = y, fill = clay.var, colour = clay.var)) +
    coord_equal()
```




## 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 ...
```

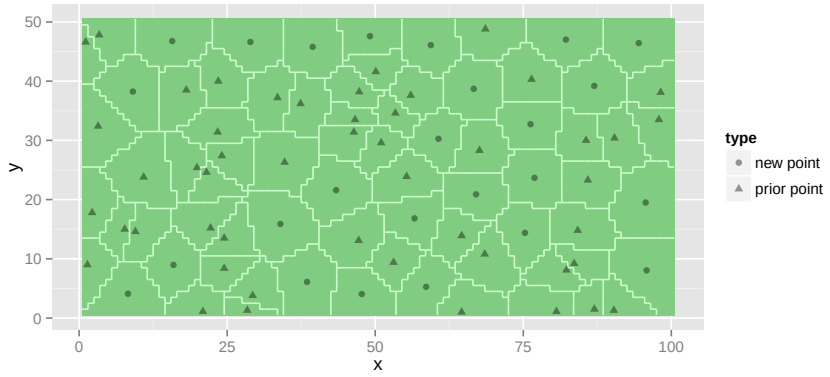
Twenty-five *new* points can be assigned to sparsely sampled regions by means of:

```
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 reduce the risk of ending up in an unfavourable local optimum.

```
plot(myStratification, mySamplingPattern)
```




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

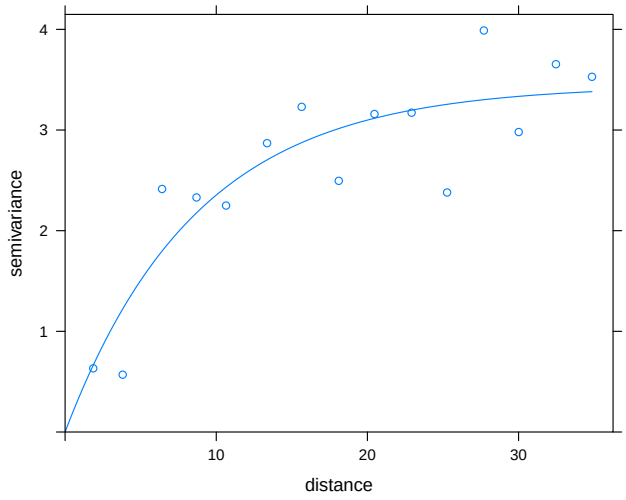
A map of clay contents can be obtained by means of ordinary point kriging. First, some field and laboratory work has to be done to obtain clay contents for the 25 new points:

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

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))
```

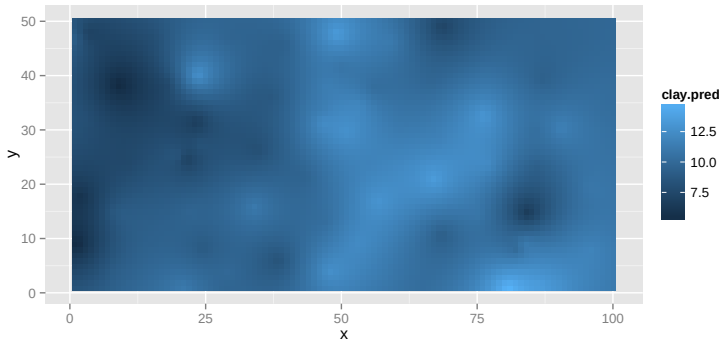
```
plot(sampleVariogram, model = variogramModel)
```



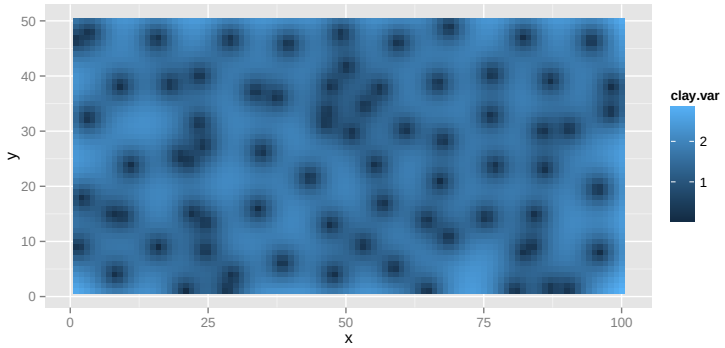
```
g <- gstat(id = "clay", formula = clay ~ 1, data = spData, model = variogramModel)
yHat <- predict(g, newdata = myAsciiGrid)


## [using ordinary kriging]
```

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

```
ggplot(data = as(yHat, "data.frame")) +
    geom_raster(mapping = aes(x = x, y = y, fill = clay.pred, colour = clay.pred)) +
    coord_equal()
```




```
ggplot(data = as(yHat, "data.frame")) +
    geom_raster(mapping = aes(x = x, y = y, fill = clay.var, colour = clay.var)) +
    coord_equal()
```

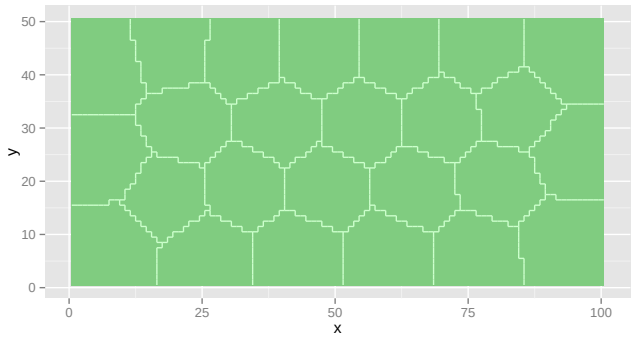



## 3.3 Random sampling from compact geographical strata

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
```

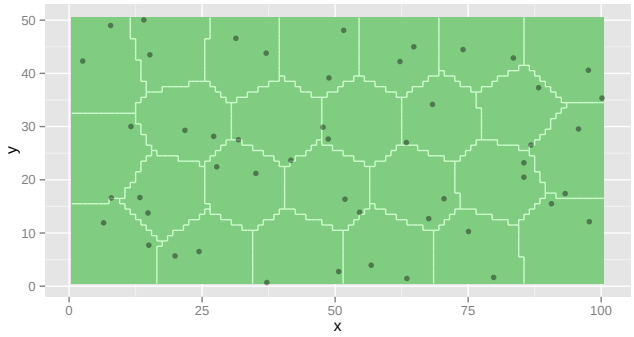
```
plot(myStratification)
```




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
```

```
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 63.49  1.449
## 2 56.78  3.943
## 3 48.71 27.650
## 4 47.73 29.880
## 5 14.99  7.705
```

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
## 10.206  5.053
```

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
## 0.2050 0.1149
```

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

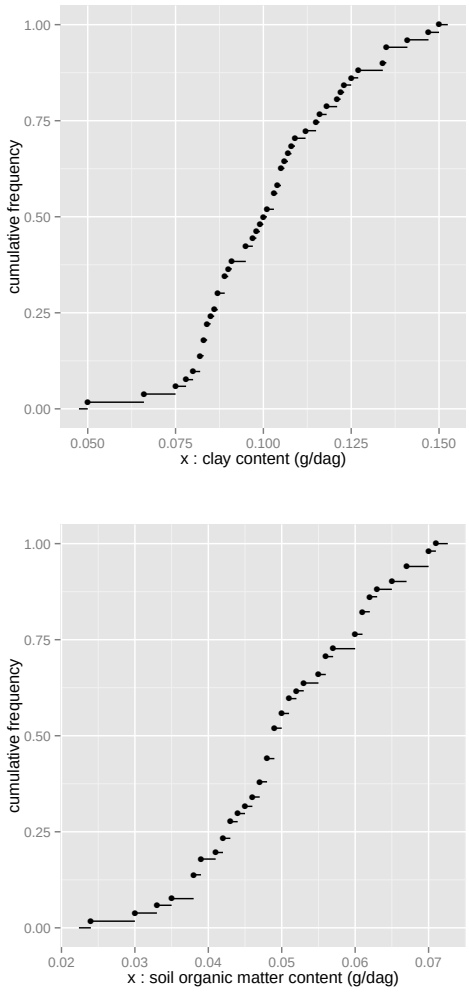
```
mySCDF <- estimate("scdf", myStratification, mySamplingPattern, myData)
```

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

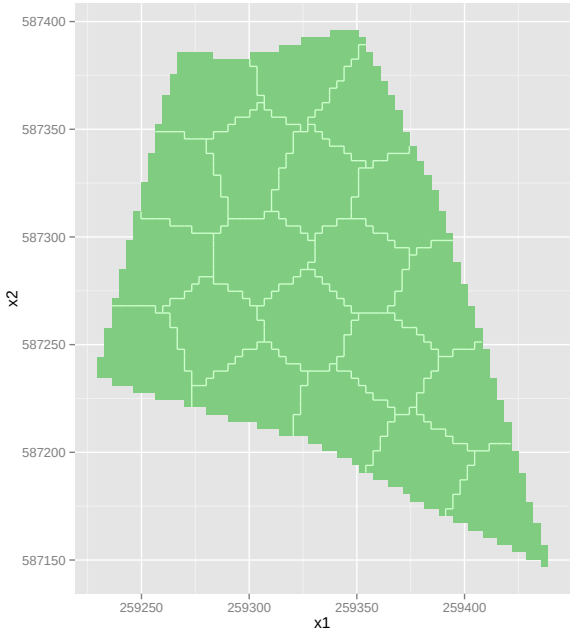
In this example the aim is to estimate the global mean clay and organic matter contents of a field. To reduce laboratory costs, the soil aliquots collected at the sampling locations will be bulked into composite samples. The study area is a field near the village of Farmsum, in the North-East of the 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., 2012):

```
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 supports 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
```

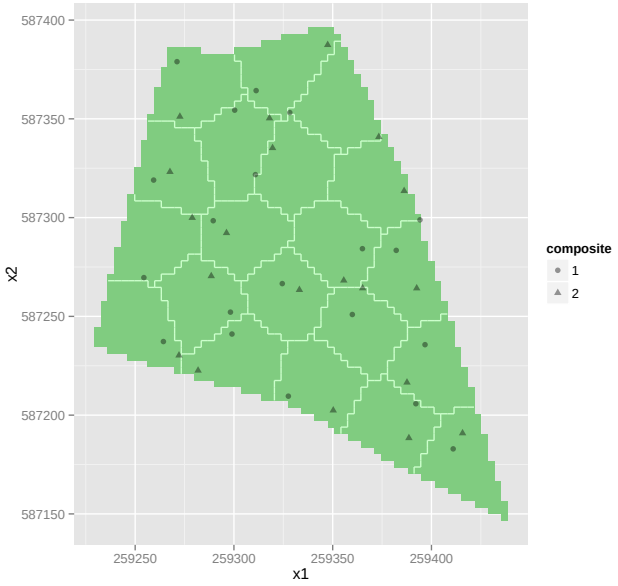
```
plot(myStratification)
```




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
```

```
plot(myStratification, mySamplingPattern)
```




Sampling points can be extracted means of:

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


##   composite     x1     x2
## 1         1 259365 587284
## 2         2 259356 587268
## 3         1 259324 587267
## 4         2 259333 587263
## 5         1 259290 587298
```

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
```

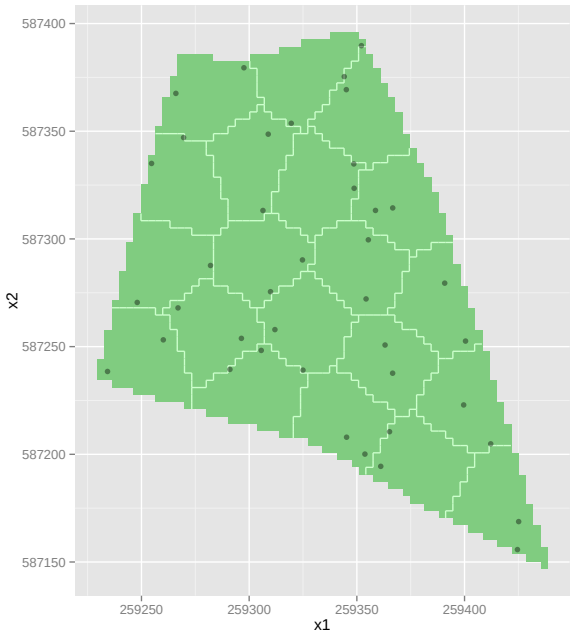
The spatial mean and its standard error can be estimated by means of:

```
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 $20 \times 2$ sampling locations:

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


##       x1     x2
## 1 259354 587272
## 2 259355 587300
## 3 259325 587239
## 4 259312 587258
## 5 259325 587290
```

```
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., 2012).

To illustrate the effect of stratification on smaller spatial scales, consider two grids covering the surface of the earth:

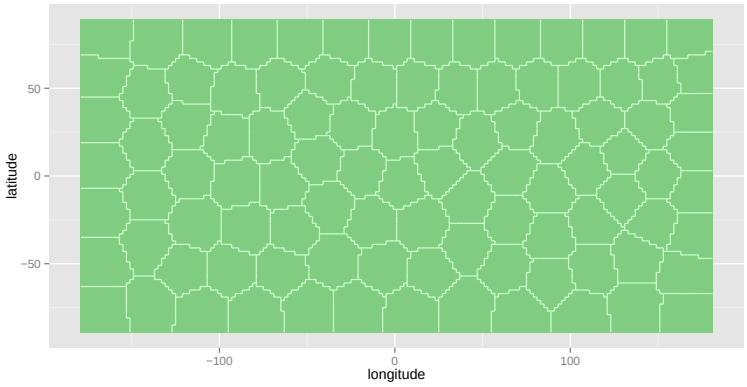
```
grd <- expand.grid(
    longitude = seq(from = -178, to = 180, by = 2),
    latitude  = seq(from =  -88, to =  88, by = 2)
)
gridded(grd) <-  ~ longitude * latitude

grd_crs <- grd
proj4string(grd_crs) <- CRS("+proj=longlat +ellps=WGS84")
```

Note that `grd` is identical to `grd_crs`, except that `grd_crs` 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.

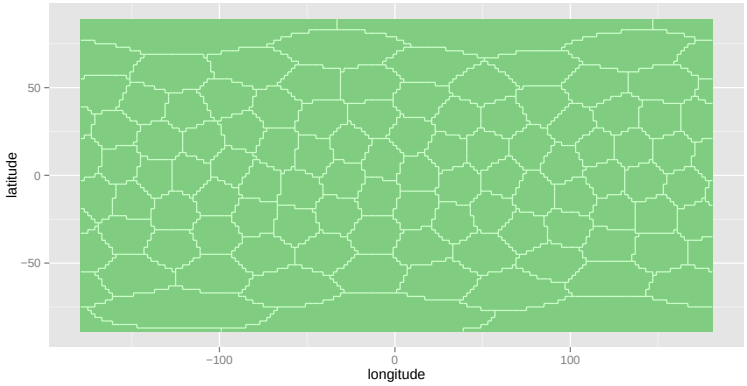
Both grids will be partitioned into 100 compact geographical strata:

```
strata     <- stratify(grd,     nStrata = 100)
strata_crs <- stratify(grd_crs, nStrata = 100)
```

```
plot(strata)
```

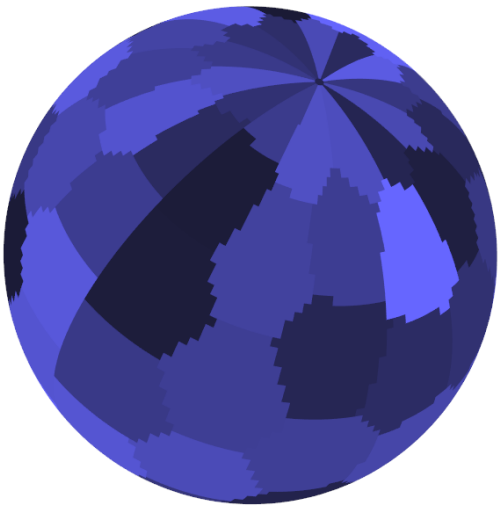



```
plot(strata_crs)
```

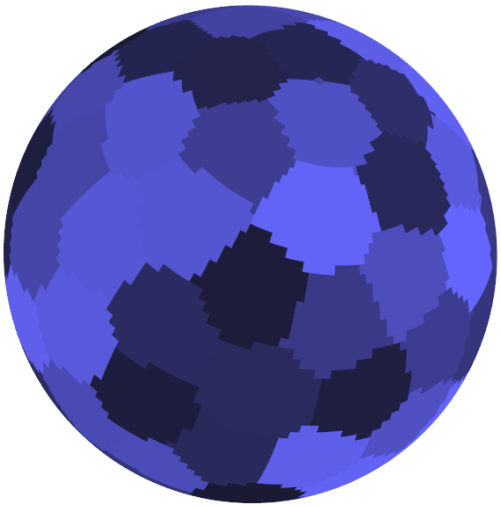



Note that `grd` seems to have more compact strata near the geographic poles than `grd_crs`. However, the contrary is true. This becomes evident when both stratifications are projected on a sphere:

```
## [1] 1
```



```
## [1] 2
```



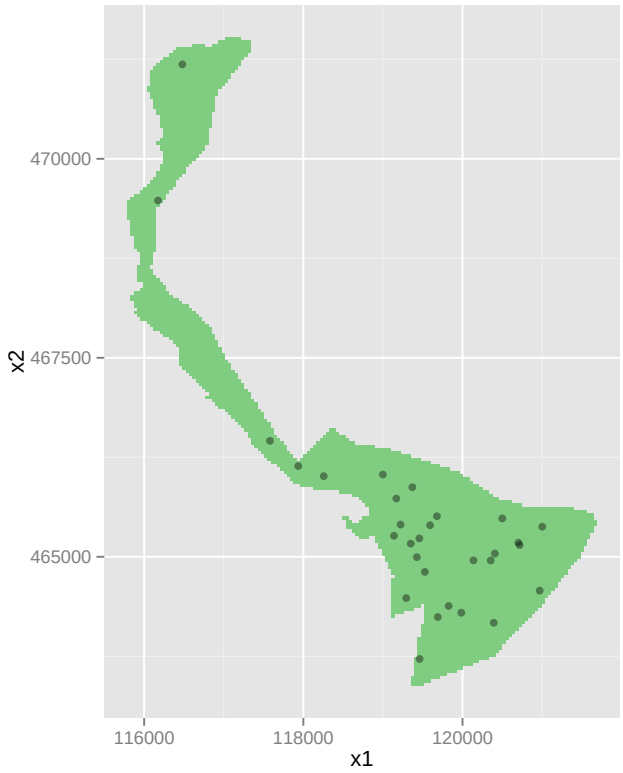
The top figure corresponds to `grd` and the bottom figure to `grd_crs`. The strata of `grd_crs` are clearly more compact than those of `grd`. In addition, `grd` suffers from pronounced edge effects near the poles and at $180^o$ longitude. The strata are discontinuous at this meridian, *i.e.*, two points on opposite sides of the meridian are treated as very distant when squared Euclidean distances are used. The strata of `grd_crs`, on the other hand, have been optimised on a sphere by using squared great circle distances, and don't suffer from edge effects, *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)
```

```
plot(myStratification, mySamplingPattern)
```




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


## 'data.frame': 30 obs. of  2 variables:
##  $ clay: num  11.8 8.2 12.5 12.4 7.6 8.9 11.3 10.8 6.4 9.2 ...
##  $ SOM : num  5.9 4.1 6 5.4 4 4.3 5.7 4.8 3.4 4.7 ...
```

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

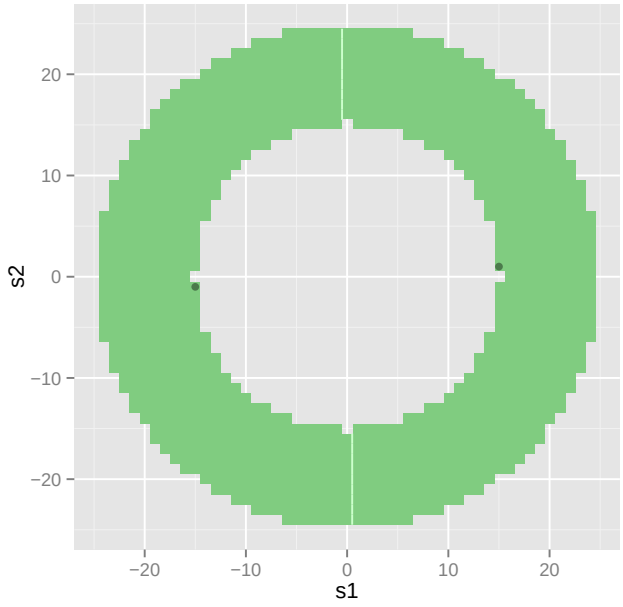
```
estimate("spatial mean", myStratification, mySamplingPattern, myData)


##   clay    SOM
## 10.103  4.963


estimate("standard error", myStratification, mySamplingPattern, myData)


##   clay    SOM
## 0.4600 0.2485
```

## 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)
```

```
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).

## 4.4 Session information

- R version 2.15.1 (2012-06-22), `x86_64-pc-mingw32`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils

- Other packages: ggplot2 0.9.2, gstat 1.0-14, knitr 0.8, rgdal 0.7-18, rgl 0.92.892, rJava 0.9-3, sp 0.9-99, spacetime 0.7-1, spcosa 0.3-1, xts 0.8-6, zoo 1.7-7
- Loaded via a namespace (and not attached): colorspace 1.1-1, dichromat 1.2-4, digest 0.5.2, evaluate 0.4.2, formatR 0.6, grid 2.15.1, gtable 0.1.1, labeling 0.1, lattice 0.20-10, MASS 7.3-21, memoise 0.1, munsell 0.3, plyr 1.7.1, proto 0.3-9.2, RColorBrewer 1.0-5, reshape2 1.2.1, scales 0.2.2, stringr 0.6.1, tools 2.15.1

# 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.

Timothy H. Keitt, Roger Bivand, Edzer Pebesma, and Barry Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2012. URL http://CRAN.R-project.org/package=rgdal. R package version 0.7-8.

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, 2012. 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.

Simon Urbanek. *rJava: Low-level R to Java interface*, 2011. URL http://CRAN.R-project.org/package=rJava. R package version 0.9-3.

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.

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*, 36:1261–1267, 2010. URL http://dx.doi.org/10.1016/j.cageo.2010.04.005.

Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book.