Tutorial: Spatially Balanced Designs in R

Roberto Benedetti
University of Chieti-Pescara, Italy

Federica Piersimoni
ISTAT, Italian National Statistical Institute, Rome, Italy

Francesco Pantalone
University of Perugia, Italy

Contents

We present some R code useful to perform the following sample designs:

- *Generalized Random Tessellation Sampling (GRTS).*
- *Spatially Correlated Poisson Sampling (SCPS).*
- *Local Pivotal Method (LPM).*

Per each of those, we selected a sample, plot it and finally compute the spatial balance index.

Data

The data used throughout the code are coming from the package `sp` (Pebesma and Bivand 2005). In particular, the dataset used is `meuse`, which contains locations and topsoil heavy metal concentrations, along with a number of soil and landscape variables at the observation locations, collected in a flood plain of the river Meuse, near the village of Stein (NL).

The total number of locations in the dataset is 155.

For the remainder, we consider these data as they were the entire population, and we sample from there.

SBI

Along the packages containing the sampling designs, we use the function `sbi` contained in the package `Spbsampling` (Pantalone, Benedetti and Piersimoni 2019) in order to compute the *Spatial Balance Index (SBI)*.

The function `sbi` has the following arguments:

- `dis`, a distance matrix NxN that specifies how far are all the pairs of units in the population;
- `pi`, a vector of first order inclusion probabilities of the units of the population;
- `s`, a vector of labels of the sample.

The output is the SBI.

Generalized Random Tessellation Sampling

The GRTS is implemented on the R package `spsurvey` (Kincaid and Olsen 2016) through the function `grts`. Instead of use it directly, we present a wrapper of it, which has the following arguments:

- `p`, a vector of length containing the inclusion probabilities of the units in the population;
- `x`, a vector of length of the first coordinates of the units in the population;
- `y`, a vector of length of the second coordinates of the units in the population.

The output is a vector of length , filled with the indicator variables of the units: 1 if selected, 0 otherwise.

Generalized Random Tessellation Sampling

```
GRTS <-  function(p, x, y)
{
  N <- length(p)
  n <-  round(sum(p))
  index <-  1:N
  s <- rep(0,times = N)
  att <-  data.frame(x = x, y = y, mdcaty =p, ids = index)
  design <-  list(None = list(panel = c(Panel1 = n), seltype =
"Continuous", caty.n = c("Caty 1" = n), over = 0))
  res <- grts(design, DesignID = "Site", SiteBegin = 1, type.frame
= "finite", src.frame = "att.frame", in.shape = NULL, sp.object =
NULL, att.frame = att, id = NULL, xcoord = "x", ycoord = "y",
stratum = NULL, mdcaty = "mdcaty", startlev = NULL, maxlev = 11,
maxtry = 1000, shift.grid = TRUE, do.sample = rep(TRUE,
length(design)), shapefile = FALSE, prjfilename = NULL, out.shape
= "sample")
  s[res$ids] <- 1
  s}
```

Generalized Random Tessellation Sampling

Through the following code we select a sample of dimension , with constant inclusion probabilities equal to .

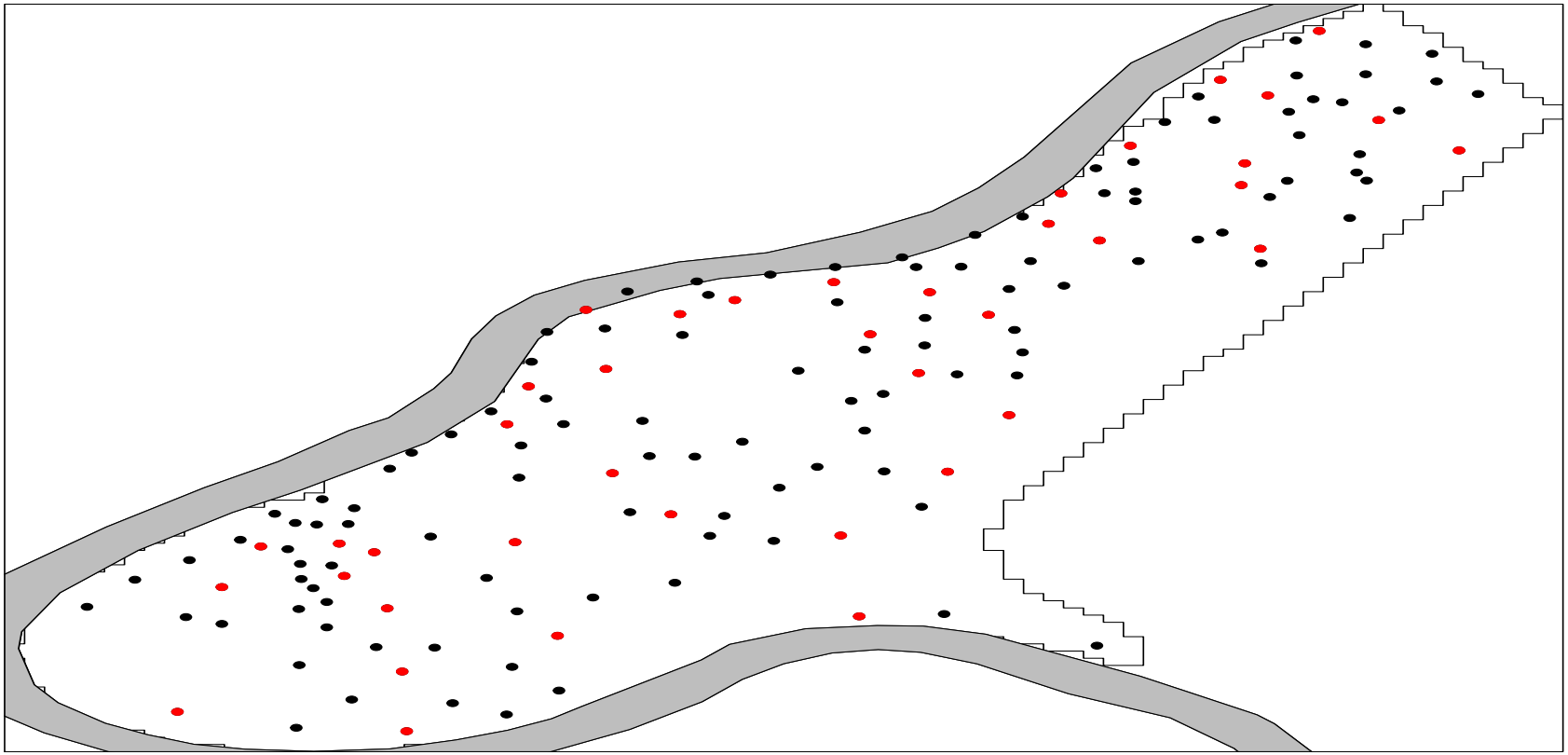
```
> N <- nrow(meuse)
> n <- 40
> pi <- rep(n / N, N)
> set.seed(42)
> grts_sample <- GRTS(p = pi, x = meuse$x, y =
  +     meuse$y)
Stratum: None
Current number of levels: 3
Current number of levels: 4
Current number of levels: 5
Final number of levels: 5
```

Generalized Random Tessellation Sampling

In order to plot the obtained sample, we use the standard function `plot`, along with `lines`, `polygon` and `points`.

```
> plot(meuse.area,axes = F, type = "l", xlab
  +     = "", ylab = "")
> lines(meuse.riv, type = "l", asp = 1)
> polygon(meuse.riv, col = "gray")
> points(meuse$x, meuse$y, pch = 19)
> points(meuse$x[grts_sample == 1], meuse
  +       $y[grts_sample == 1], pch = 19, col
  +       = "red")
> box()
```

Generalized Random Tessellation Sampling



The black points are all the units in the population, red points are the selected unit in the sample.

Generalized Random Tessellation Sampling

Finally, we compute the SBI.

```
> d <- as.matrix(dist(cbind(meuse$x, meuse$y)))
> sbi(d, pi, which(grts_sample == 1))
[1] 0.1987246
```

Spatially Correlated Poisson Sampling

The SCPS is implemented on the package `BalancedSampling` (Grafstrom 2016).

In particular, the method is implemented by means of the function `scps`, which has the following two arguments:

- `prob`, which is a vector of length N with the inclusion probabilities;
- `x`, which is the $N \times q$ matrix of auxiliary variables.

The output of the function is a vector of length n containing the labels of the selected units in the sample.

Note that, if the inclusion probabilities sum to n, then the sample size is fixed to n.

Spatially Correlated Poisson Sampling

We select a sample of dimension and equal inclusion probabilities , using as matrix of covariates the coordinates of the units, as in the follow.

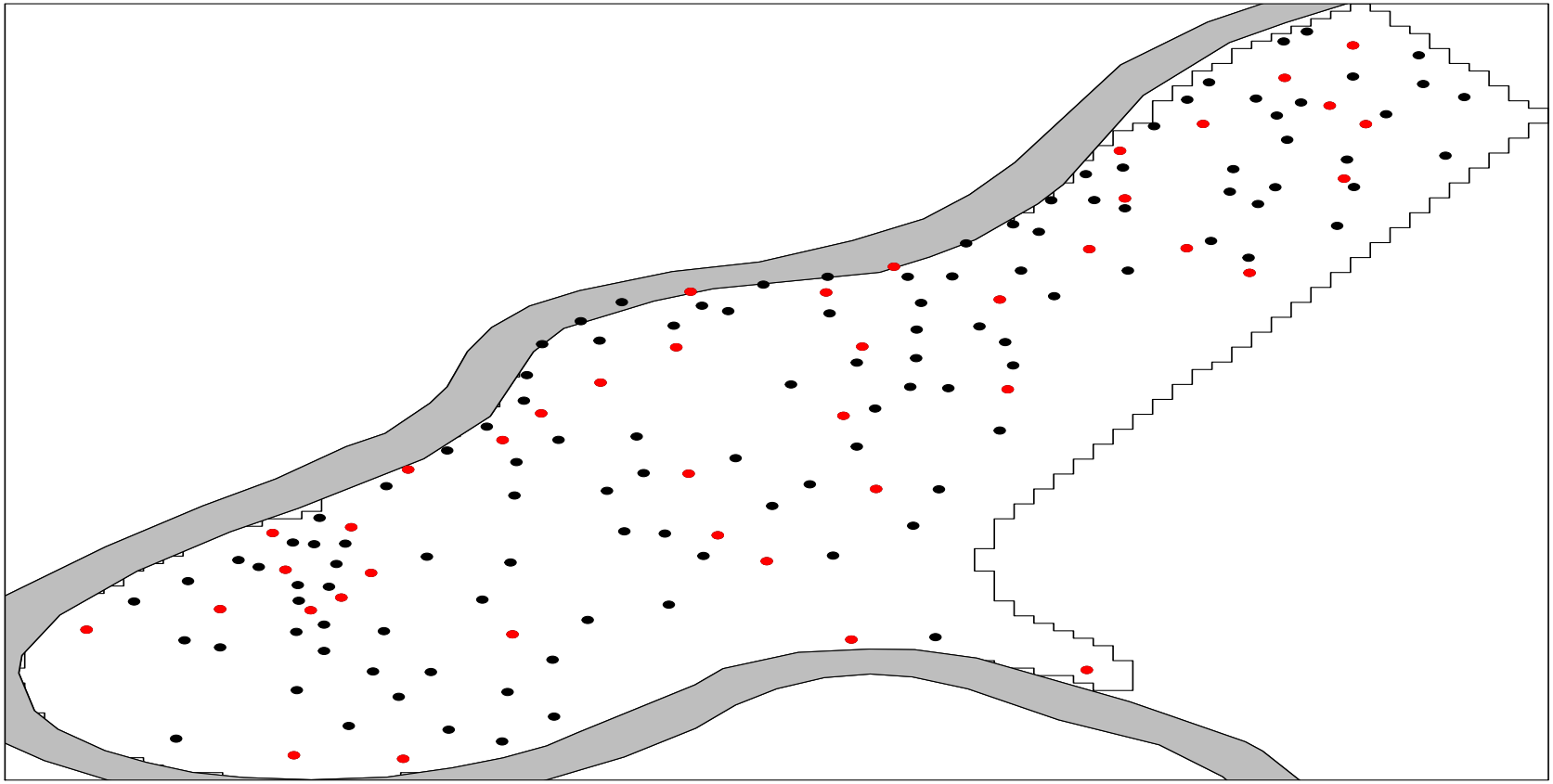
```
> N <- nrow(meuse)
> n <- 40
> pi <- rep(n / N, N)
> X <- cbind(meuse$x, meuse$y)
> set.seed(42)
> scps_sample <- scps(pi, X)
```

Spatially Correlated Poisson Sampling

We obtain the plot through

```
> plot(meuse.area,axes = F, type = "l",
+      xlab = "", ylab = "")
> lines(meuse.riv, type = "l", asp = 1)
> polygon(meuse.riv, col = "gray")
> points(meuse$x, meuse$y, pch = 19)
> points(meuse$x[scps_sample], meuse
+     $y[scps_sample], pch = 19, col =
+     "red")
> box()
```

Spatially Correlated Poisson Sampling



The black points are all the units in the population, red points are the selected unit in the sample.

Spatially Correlated Poisson Sampling

The spatial balance index is

```
> d <- as.matrix(dist(cbind(meuse$x, meuse$y)))
> sbi(d, pi, scps_sample)
[1] 0.1338349
```

Local Pivotal Method

The LPM is implemented on the package `BalancedSampling` (Grafstrom 2016).

In particular, the two algorithms coming from this method are implemented by means of the function `lpm1` and `lpm2`. Both function share the same following major arguments:

- `prob`, which is a vector of length N with the inclusion probabilities;
- `x`, which is the $N \times q$ matrix of auxiliary variables.

The output for both functions is a vector of length n containing the labels of the selected units in the sample.

Note that, if the inclusion probabilities sum to n, then the sample size is fixed to n.

Local Pivotal Method

As we did before for the SCPS, we use as matrix of the covariates the coordinates of the units, and we select a sample of dimension with equal inclusion probabilities , this time selecting one sample by means of LPM1 and one sample by means of LPM2.

```
> N <- nrow(meuse)
> n <- 40
> pi <- rep(n / N, N)
> X <- cbind(meuse$x, meuse$y)
> set.seed(42)
> lpm1_sample <- lpm1(pi, X)
> set.seed(42)
> lpm2_sample <- lpm2(pi, X)
```

Local Pivotal Method

The plots are produced by

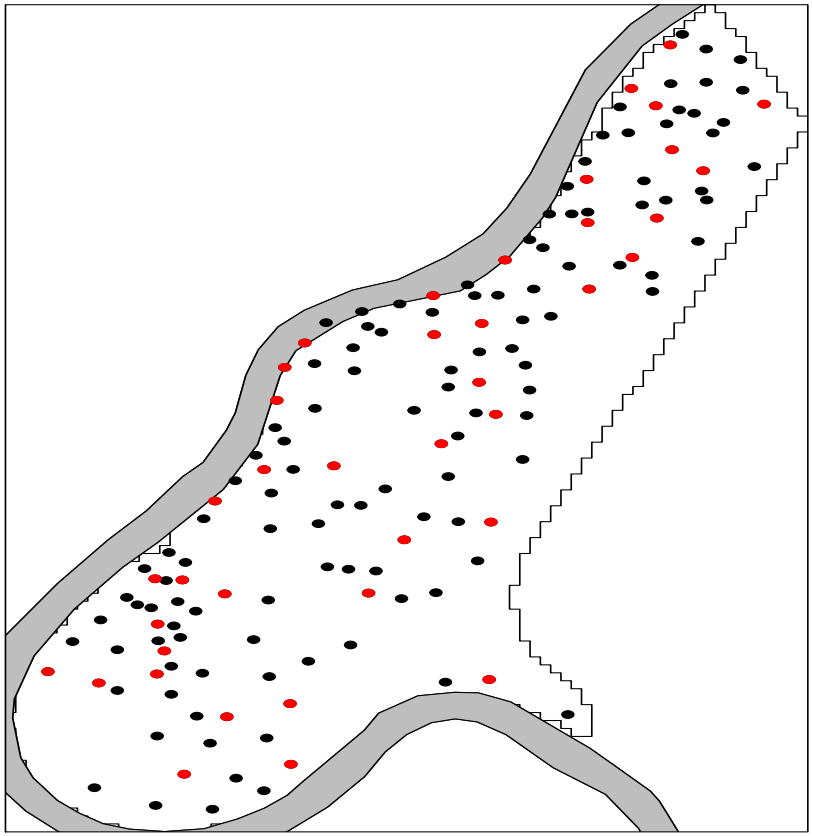
```
> par(mfrow = c(1, 2))
> plot(meuse.area, axes = F, type = "l",
xlab = "", ylab = "")
> lines(meuse.riv, type = "l", asp = 1)
> polygon(meuse.riv, col = "gray")
> points(meuse$x, meuse$y, pch = 19)
> points(meuse$x[lpm1_sample], meuse
$y[lpm1_sample], pch = 19, col = "red")
> box()
```

Local Pivotal Method

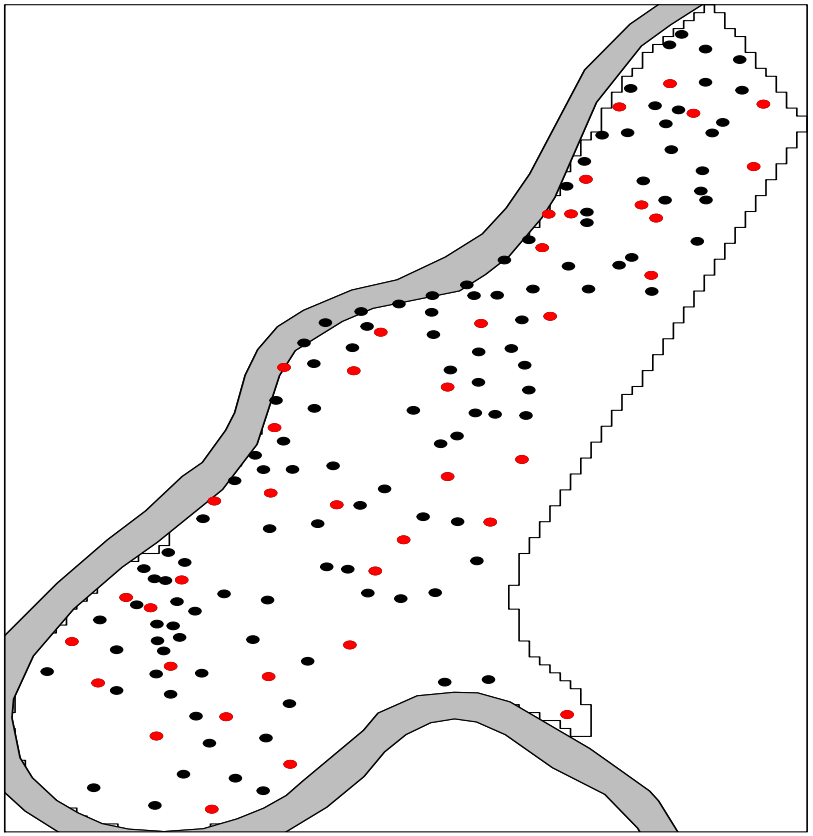
```
> plot(meuse.area, axes = F, type = "l",
xlab = "", ylab = "")
> lines(meuse.riv, type = "l", asp = 1)
> polygon(meuse.riv, col = "gray")
> points(meuse$x, meuse$y, pch = 19)
> points(meuse$x[lpm2_sample], meuse
$y[lpm2_sample], pch = 19, col = "red")
> box()
```

Local Pivotal Method

LPM1



LPM2



The black points are all the units in the population, red points are the selected unit in the sample.

Local Pivotal Method

The respective spatial balance indexes for these samples are computed by

```
> d <- as.matrix(dist(cbind(meuse$x, meuse
$y)))
> sbi(d, pi, lpm1_sample)
[1] 0.1372502
> sbi(d, pi, lpm2_sample)
[1] 0.2328771
```

Thank you for your attention!