Tutorial: Spbsampling

Roberto Benedetti
University of Chieti-Pescara, Italy

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

Francesco Pantalone
University of Perugia, Italy

Introduction

The R package `Spbsampling` (Pantalone, Benedetti and Piersimoni 2019) provides function for spatially balanced sampling. In particular, the implemented sampling designs are PWD, SWD and HPWD, by means of the functions `pwd`, `swd`, and `hpwd`, respectively.

Along these functions, there are other complementary ones.

Structure of the presentation:

- firstly, we introduce functions designed for standardization of matrix and spatial balance index, since they are needed successively;
- secondly, we present the functions for the design, accompanied by R examples;
- finally, we discuss a case-study for the effect of the β parameter.

Matrix standardization

The function `stprod` and `stsum` are designed for the standardization of the distance matrix. They require the following parameters:

- `mat`: distance matrix to be standardized (of dimension N × N);
- `vec`: constraints, of length N (because we have a constraint for each row/column of the distance matrix);
- `differ`: maximum accepted difference from the constraints, set equal to 1e-15;
- `niter`: maximum number of iterations, set equal to 1000.

The output consists on the standardized matrix.

SBI

The function `sbi` compute the Spatial Balance Index and 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.

Data

The data used throughout the first part of the presentation are coming from the dataset `meuse` of the package `sp` (Pebesma and Bivand 2005). It regards 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.

Product Within Distance (PWD)

The function `pwd` requires the following parameters:

- `dis`: distance matrix of the target population (of dimension $N \times N$);
- `nsamp`: sample size;
- `nrepl`: number of samples to draw, set to default at 1;
- `niter`: number of iterations for the algorithm, set to default at 10.

The parameter is regulated by the exponent of the distance matrix $\boldsymbol{D}$, in the following way $\boldsymbol{D}\uparrow x \Rightarrow \beta = x$.

The output of the function is a matrix `nrepl` x `nsamp`, which contains the `nrepl` selected samples, each of them stored in a row. In particular, the i-th row contains all labels of units selected in the i-th sample.

Product Within Distance (PWD)

In order to achieve equal inclusion probabilities $\pi\downarrow i = n/N$ $\forall i \in U$, we need to standardize the distance matrix.

We do this with the following code

```
> d <- as.matrix(dist(cbind(meuse$x, meuse
$y)))
> s_d <-stprod(mat = d^10, vec = rep(0, N))
```

Now, we can use it as input for the function `pwd`

```
> N <- nrow(meuse)
> n <- 40
> set.seed(42)
> pwd_sample <- pwd(dis = s_d, nsamp = n)
```

Product Within Distance (PWD)

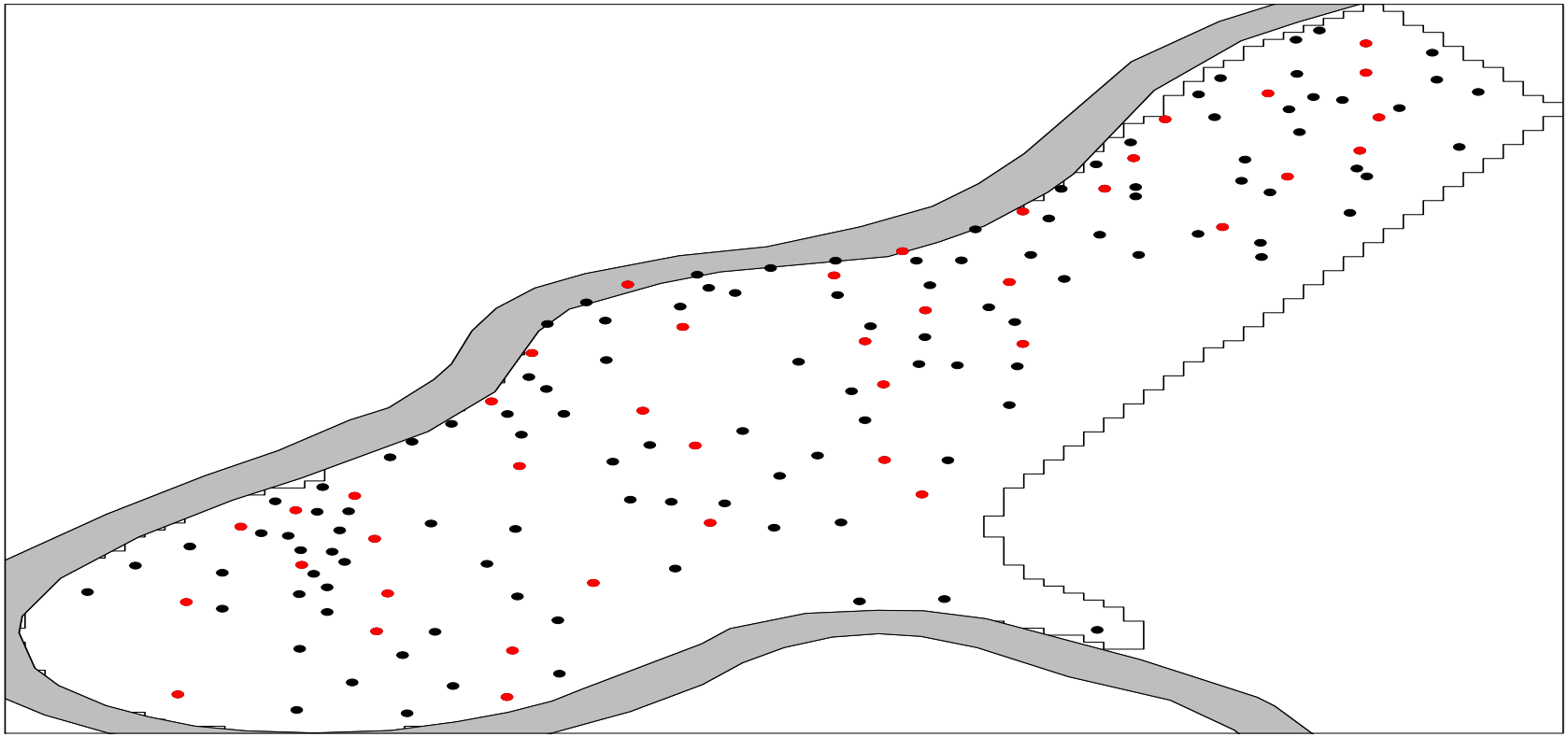
The spatial balance index can be computed by

```
sbi(dis = d, pi = rep(n / N, N), s = pwd_sample)
[1] 0.092852
```

and the plot of the selected sample is achieved by

```
> 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[pwd_sample], meuse
$y[pwd_sample], pch = 19, col = "red")
> box()
```

Product Within Distance (PWD)



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

Sum Within Distance (SWD)

The function `swd` shares all the parameters with the function `pwd`, and has an additional one:

- `bexp`: distance matrix of the target population (of dimension).

Therefore, in this case, in order to set the value of we do not have to modify the power of the distance matrix as in the `pwd` case.

The output of the function is the same as before.

Sum Within Distance (SWD)

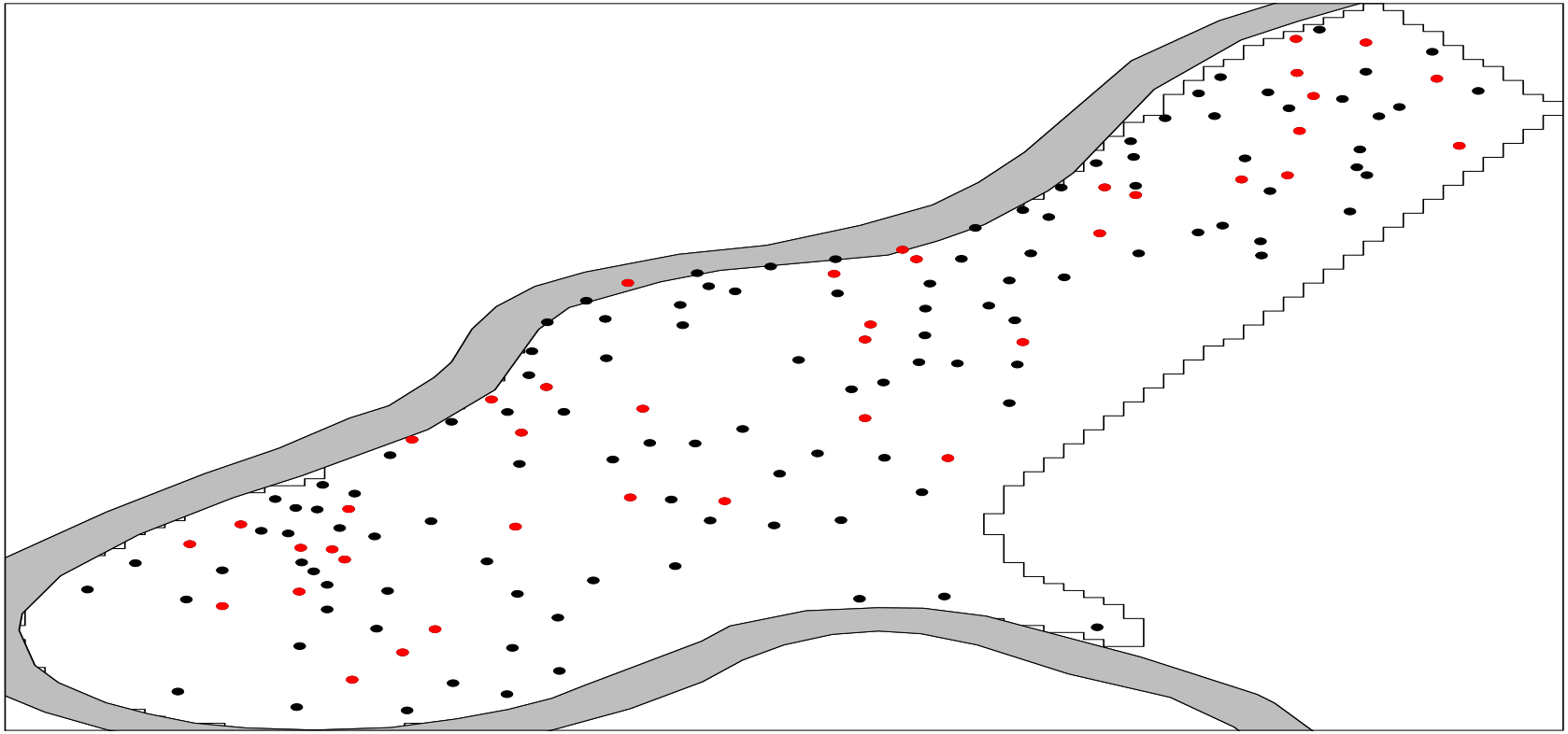
As before, we need to standardize the distance matrix in order to achieve equal inclusion probabilities, but this time we need to use the function `stsum`, specifically designed for the SWD algorithm.

```
> d <- as.matrix(dist(cbind(meuse$x, meuse$y)))
> s_d <-stsum(mat = d, vec = rep(1, N))
```

Then, we select a sample with a value of beta = 10, set through the parameter `bexp`

```
> set.seed(42)
> swd_sample <- swd(dis = s_d, nsamp = n, bexp = 10)
```

Sum Within Distance (SWD)



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

Heuristic Product Within Distance (HPWD)

The function `hpwd` uses only some of the parameters seen for the `pwd`. In particular, the distance matrix `dis`, the sample size `nsamp` and the number of samples `nrepl`.

Note that the parameter β is regulated as in the case of `pwd`, i.e. by the power of the distance matrix.

The output is as the previous cases.

As always, we need to standardize the matrix in order to achieve equal inclusion probabilities. For this sampling design, we need to use the function `stprod`

```
> d <- as.matrix(dist(cbind(meuse$x, meuse
$y)))
> s_d <-stprod(mat = d^10, vec = rep(0, N))
```

Heuristic Product Within Distance (HPWD)

We select a sample of dimension

```
> N <- nrow(meuse)
> n <- 40
> set.seed(42)
> hpwd_sample <- hpwd(dis = s_d, nsamp = n)
```

and compute the Spatial Balance Index

```
> sbi(dis = d, pi = p, s = hpwd_sample)
[1] 0.07919101
```

Effect of β

We use a simulated population in order to have a glance to the effect that has on the selection of the sample. Specifically, we use the dataset `simul2` from the package `Spbsampling` and we select one sample per each value of .

`simul2` is a simulated georeferenced population of dimension . The coordinates are generated in the range as a simulated realization of a particular random point pattern: the Neyman-Scott process with Cauchy cluster kernel.

Effect of β

Samples selection:

```
> N <- nrow(simul2)
> n <- 100
> p <- rep(n / N, N)
> d <- as.matrix(dist(cbind(simul2$x, simul2$y)))
> s_d1 <-stprod(mat = d^1, vec = rep(0, N))
> set.seed(42)
> pwd1_sample <- pwd(dis = s_d1, nsamp = n)
> sbi(dis = d, pi = p, s = pwd1_sample)
[1] 0.2547475
```

Effect of β

```
> s_d5 <-stprod(mat = d^5, vec = rep(0, N))
> set.seed(42)
> pwd5_sample <- pwd(dis = s_d5, nsamp = n)
> sbi(dis = d, pi = p, s = pwd5_sample)
[1] 0.0810101
> s_d10 <-stprod(mat = d^10, vec = rep(0, N))
> set.seed(42)
> pwd10_sample <- pwd(dis = s_d10, nsamp = n)
> sbi(dis = d, pi = p, s = pwd10_sample)
[1] 0.0589899
```

Effect of β

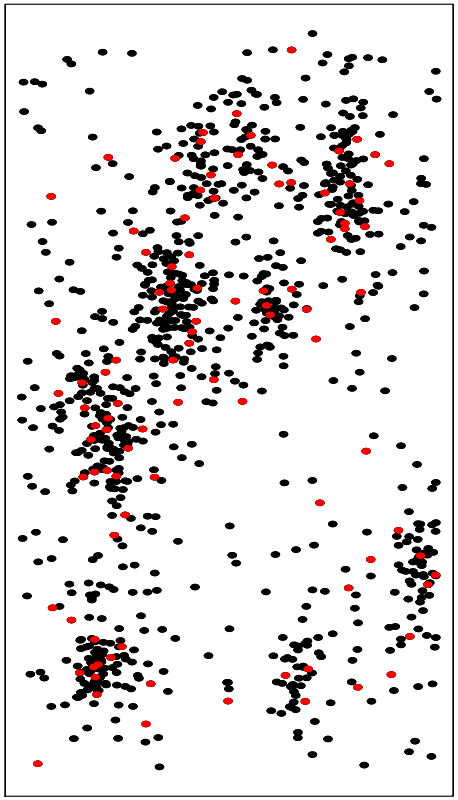
Plots:

```
> par(mfrow=c(1, 3))
> plot(simul2$x, simul2$y, axes = F, xlab = "", ylab
= "")
> points(simul2$x, simul2$y, pch = 19)
> points(simul2$x[pwd1_sample],
simul2$y[pwd1_sample], pch = 19, col = "red")
> box()
> plot(simul2$x, simul2$y, axes = F, xlab = "", ylab
= "")
> points(simul2$x, simul2$y, pch = 19)
> points(simul2$x[pwd5_sample],
simul2$y[pwd5_sample], pch = 19, col = "red")
> box()
```

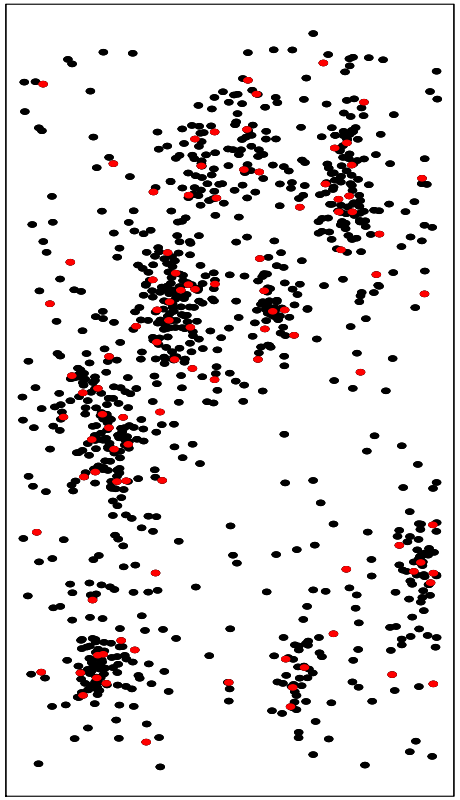
Effect of β

```
> plot(simul2$x, simul2$y, axes = F, xlab = "", ylab
= "")
> points(simul2$x, simul2$y, pch = 19)
> points(simul2$x[pwd10_sample],
simul2$y[pwd10_sample], pch = 19, col = "red")
> box()
```

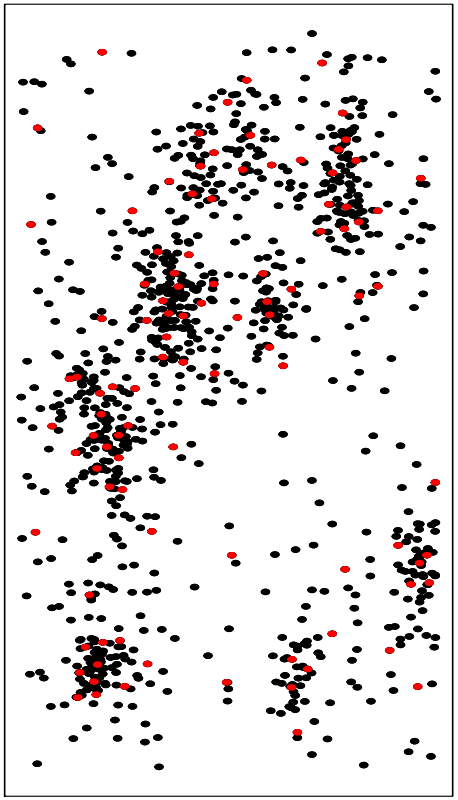
Effect of β



β=1, sbi ≈ 0.25



β=5, sbi ≈ 0.08



β=10, sbi ≈ 0.05

Future research

Did it but not yet included in the package because the algorithms are not yet published

- Balanced Sampling
- Doubly Balanced Sampling
- Coordination Balanced Samples

Research topics

- Possibility to efficiently achieve unequal inclusion probabilities.
- Variance estimation (explicit or resampling solutions)
- Continuous populations (line-point transects)

Interesting topics but not addressed even though they would be important to add to the package

- Use of dependence in the estimation phase (not difficult if it is model-based but more complex if model-assisted or even design-based)

Thank you for your attention!