Using Durbach et al. optimization in secrdesign

Murray Efford and Ian Durbach

2022-10-07

Contents

Introduction

IN PREPARATION

This is a demonstration of the min(n,r) method of Durbach et al. (2021) for optimizing the placement of detectors in a spatially explicit capture–recapture study.

Preliminaries

```
library(secrdesign)
library(sf)          # spatial functions
```

Some messages have been suppressed.

The arguments of GAminnr mostly follow the description in Durbach et al. (2021) Appendix F. Some argument names have been altered for consistency with **secrdesign**

```
GAminnr <- function (mask, alltraps, ntraps, detectpar, noccasions = 1,      detectfn =
c("HHN", "HHR", "HEX", "HAN", "HCG"), D = NULL, penalty = NULL,      seed = NULL, ...)
```

Argument	Default	Description
mask	*	habitat mask, including the buffer region. Can be provided as a secr mask object or as a matrix, in which case it is converted to a secr mask internally
alltraps	*	two-column dataframe comprising the x- and y-coordinates of all possible detector locations
ntraps	*	number of detectors in required design
detectpar	*	list of detection parameters lambda0, sigma

Argument	Default	Description
noccasions	*	number of survey occasions
detectfn	'HHN'	detection function
D	NULL	if supplied, either a constant, specifying the expected number of activity centres per ha, or a vector of length M, the number of cells in the mask. If a vector, values reflect expected densities per ha in each mask cell, and the order of values in D should be the same as the order of cells in the mask
penalty	NULL	list describing penalty (see separate section)

* indicates no default - a value must be provided.

Other named values are passed to `kofnGA` in the dots argument, most importantly:

Argument	Default	Description
ngen	500	number of generations to run
popsize	200	size of the population; equivalently, the number of offspring produced each generation
verbose	0	integer controlling the display of progress during search. If a positive value, then the iteration number and best objective function value are displayed at the console every 'verbose' generations. Otherwise nothing is displayed. The default gives no display.
cluster	NULL	number of parallel cores or a prebuilt parallel cluster

The detector type ("count", "proximity", or "multi") is inferred from `alltraps`.

This implementation does not allow lambda0 to vary. In the original, lambda0 could be a vector of length N, the number of possible detector locations. Values then are the spatially-varying expected encounter rates from each possible detector location per survey occasion, and the order of values in lambda should be the same as the order of cells in the alltraps object.

The optimal design is not affected by multiplication of D by a constant, since whatever design maximizes min(E(n), E(r)) also maximizes min(E(n), E(r)), and so D does not have to be specified (a default of 1 is used). However, estimates of E(n) and E(r) are only correct if densities are correctly specified

Example from Durbach et al. Appendix F

First prepare some inputs:

```
# input file TostExample.RData contains the data frames:
#   bigmask_df -- mask locations
#   alltraps_df -- all potential camera locations (subset of mask)
load(system.file("example", "TostExample.RData", package = "secrdesign"))
mask <- read.mask(data = bigmask_df)
alltraps <- read.traps(data = alltraps_df, detector = "count")
detectpar <- list(lambda0 = 1, sigma = 3000)
D <- 2/10000    # about twice snow leopard density
set.seed(1237)
```

Next generate an optimal design (50 generations with pop size 1000 was used by Durbach et al. 2021):

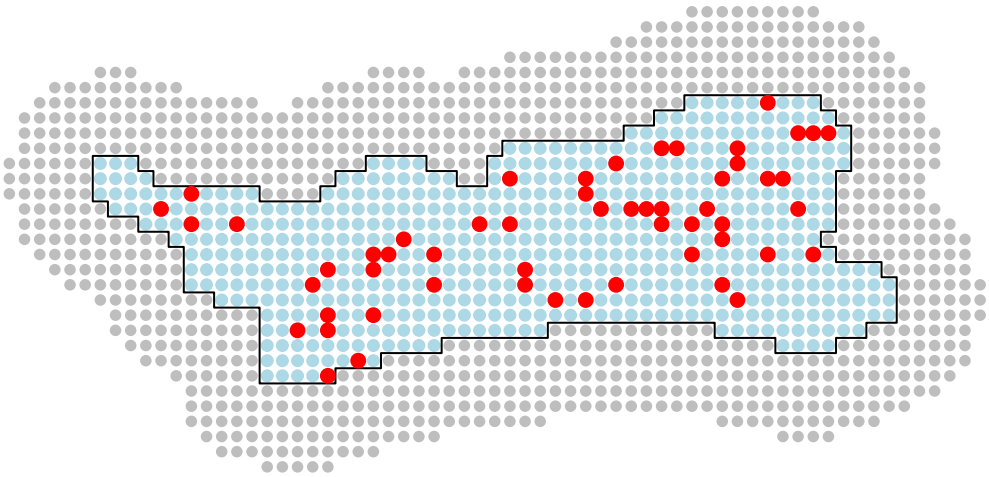
```
mnr <- GAminnr(
    mask       = mask,
    alltraps   = alltraps,   # possible positions
```

```
    ntraps     = 55,          # number of cameras to be placed
    detectpar  = detectpar,
    detectfn   = 'HHN',
    D          = D,
    noccasions = 1,
    ngen       = 50,
    popsize    = 500)
```

It's handy to make a polygon object for the region within which traps were placed:

```
region <- st_union(gridCells(alltraps))
```

```
plot(mask)
plot(alltraps, detpar=list(pch=16, cex=0.9, fg = 'lightblue'), add = TRUE)
plot(region, add = TRUE)
plot(mnr$optimaltraps, add = TRUE, detpar = list(pch=16, fg='red', cex=1.1))
```



Other designs may be compared:

```
set.seed(123)
random55 <- subset(alltraps, sample.int(nrow(alltraps), 55))

set.seed(123)
lacetrps <- make.lacework(region=region, spacing=c(20000, 4000), rotate = 45, detector = 'count')

set.seed(123)
syst55 <- make.systematic(spacing = 6500, region=region, detector = 'count')
nrow(syst55)
```

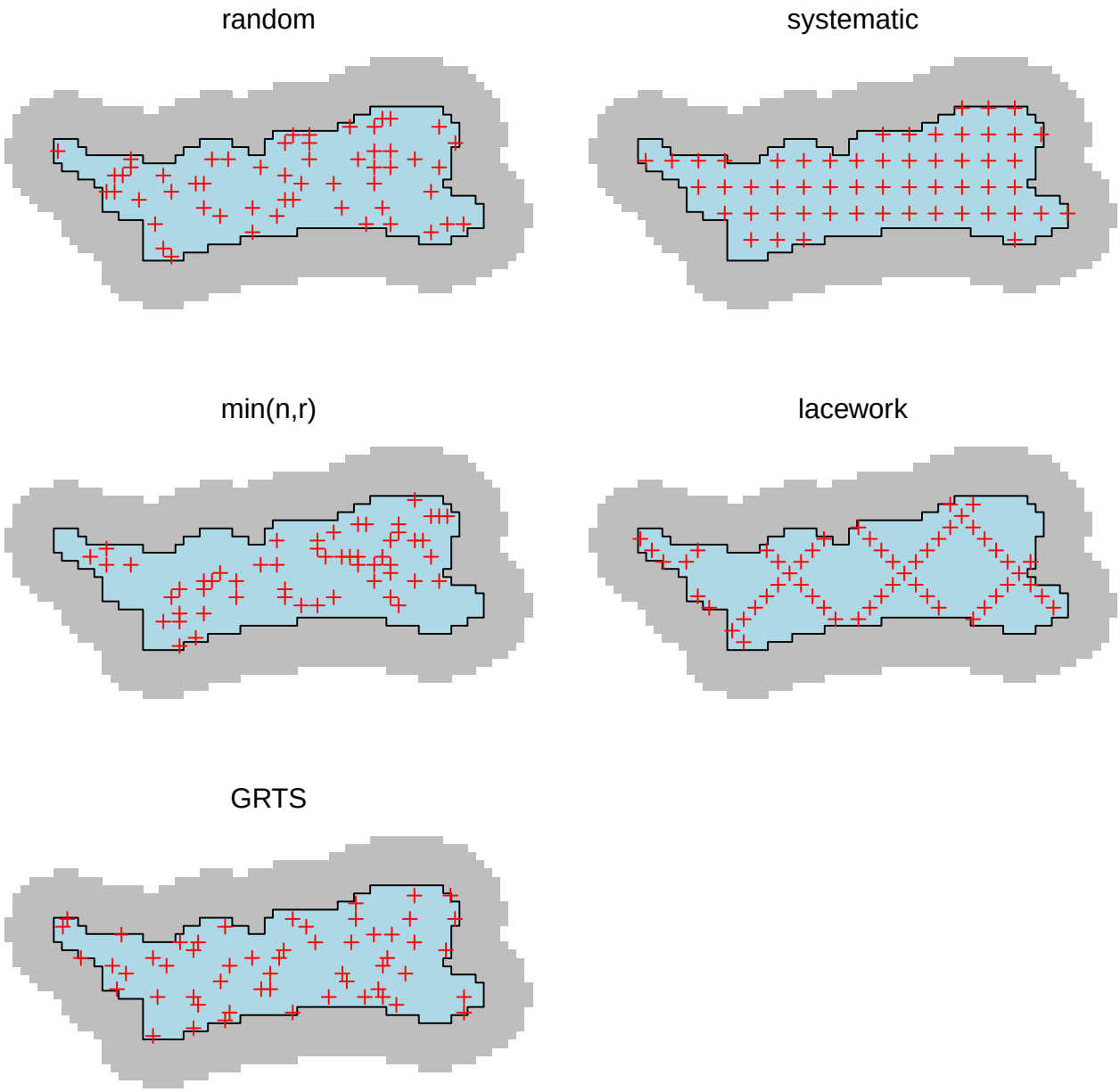
```
## [1] 55
```

```
grts <- trap.builder(n = 55, region = region, method = "GRTS", detector = "count")

plotone <- function(traps, title) {
    plot(mask, dots = FALSE)
    plot(region, col = 'lightblue', add=T)
    plot(traps, add = T)
    mtext(side = 3, title)
}

par(mfrow = c(3,2), mar = c(1,1,3,1))
```

```
plotone(random55, 'random')
plotone(syst55, 'systematic')
plotone(mnr$optimaltraps, 'min(n,r)')
plotone(lacetrps, 'lacework')
plotone(grts, 'GRTS')
```




```
traplist <- list(random55, mnr$optimaltraps, lacetrps, syst55, grts)
names(traplist) <- c("random55", "mnr$optimaltraps", "lacetrps", "syst55", "grts")
enrm <- t(sapply(traplist, Enrm, D = D, mask = mask, noccasions = 1,
    detectpar = detectpar, detectfn = 'HHN'))
enrm <- data.frame(enrm)
enrm$minnrRSE <- apply(1/sqrt(enrm[,1:2]),1,max)
enrm
```

```
##                        En       Er       Em  minnrRSE
```

```
## random55         35.65507 26.54829 16.41671 0.1940804
## mnr$optimaltraps 31.10173 31.10171 21.95239 0.1793114
## lacetrps         34.48778 27.71552 17.87097 0.1899497
## syst55           38.04885 24.15423 13.20645 0.2034714
## grts             37.34595 24.85697 14.33130 0.2005746
```

Penalty term (optional)

Undesirable designs are avoided by penalizing those that have fewer moderately close detectors than a sample from a regular grid (Durbach et al. 2021). No penalty is applied by default, and it is unclear exactly when a penalty is beneficial.

The penalty is defined by the 'penalty' list argument of 'GAminnr' with components as follows:

Component	Description
pen_wt	a weight greater than zero; usually set to a large value such as 100
pen_gridsigma	multiple of sigma used to define reference grid
pen_fn	function with arguments 'traps' and 'sigma' used to define penalty (default `GApenfn`)

The reference grid is a grid of detectors at spacing pen_gridsigma * sigma. The function pen_fn is used initially to compute a vector g of reference values from `ntraps` detectors forming a compact subset of the reference grid. At each evaluation of the objective function a penalty is calculated by comparing the reference vector to a new vector v of values from applying pen_fn to a candidate array.

The penalty is $p = penwt \sum_{j=1}^{J} max(0, g_j - v_j)$. The penalty increases whenever the reference value exceeds the observed value. The default pen_fn has $J = 2$ and returns the number of detector pairs separated by 2.5–3.5σ and 3.5–4.5σ. The objective to be minimized is then $p - min(E(n), E(r))$.

Review

Algorithmic placement methods start with a fixed number of detectors to be placed.

A fixed number of detectors may also be placed at random or by spatially balanced methods such as GRTS.

For systematic designs that use a random origin the number is a random variable.

References

Durbach, I., Borchers, D., Sutherland, C. and Sharma, K. (2021) Fast, flexible alternatives to regular grid designs for spatial capture–recapture. *Methods in Ecology and Evolution* **12**}, 298–310. DOI 10.1111/2041-210X.13517

Efford, M. G., and Boulanger, J. (2019) Fast evaluation of study designs for spatially explicit capture-recapture. *Methods in Ecology and Evolution* **10**, 1529–1535.

Wolters, M. A. (2015) A genetic algorithm for selection of fixed-size subsets with application to design problems. *Journal of Statistical Software, Code Snippets*, **68**, 1–18. DOI 10.18637/jss.v068.c01

Appendix. Fast, flexible alternatives to regular grid designs for spatial capture-recapture.

[Readme.md file from Durbach et al. (2021) Supplements Appendix F. https://github.com/iandurbach/opti

mal-secr-design/oSCR]

Code for choosing detector locations for SCR surveys that maximize the approximate precision of density estimators. Detectors are placed to maximize whichever is the smaller of E(n), the number of animals detected (first captures), and E(r), the number of recaptures (total detections less first captures), based on the approximation in Efford and Boulanger (2019): CV(Dhat) ~ 1/sqrt[min{E(n),E(r)}]. The resulting designs are called min(n,r) designs. The code and output in this repo accompany the paper

- Durbach, I., Borchers, D., Sutherland, C., & Sharma, K. Fast, flexible alternatives to regular grid designs for spatial capture-recapture. To appear in Methods in Ecology and Evolution.

Functions for constructing min(n,r) survey designs have been developed by combining two core design functions *scrdesignGA* and *scrdesignOF* from the R package *oSCR* with the *Enrm* function from the *secrdesign* package, which provides fast C implementation of the E(n) and E(r) calculations. Calculations for uniform and spatially-varying density can be handled by *secrdesign*'s *Enrm* function directly. For spatially-varying detection, we provide a modified version of this function – this requires compiling the underlying C code. The *scrdesignGA* function in *oSCR* is essentially a wrapper function around the *kofnGA* function in package *kofnGA*, which implements a genetic algorithm.

Our design functions, renamed to *scrdesignGAenr* and *scrdesignOFenrm*, are available in the **oSCR** folder in this repository. The functions require the *secrdesign* package, primarily because the calculations of E(n) and E(r) use *secr* mask objects. Function *scrdesignOFenrm* is called by *scrdesignGAenrm* and requires no input from the user – it has simply been updated to calculate the number of first captures and recaptures respectively. Function *scrdesignGAenrm* requires a number of inputs, these are described in the supplementary material of the paper above, or in the comments to the *scrdesignGAenrm.R* file in the *oSCR* folder.

The only files in this repo that are strictly required to generate min(n,r) designs are *scrdesignGAenrm.R* and *scrdesignOFenrm.R* in the **oSCR** directory. To use these, download the files into your project directory and source them with:

```
source(\my_project_dir\scrdesignGAenrm.R)
source(\my_project_dir\scrdesignOFenrm.R)
```

Required packages are *dplyr*, *sf*, *secrdesign*, and *oSCR*. For a simple example generating min(n,r) designs for uniform and non-uniform activity centre densities see *example-uniformD.R* and *example-spatialD.R* respectively.

Running designs with spatially-varying *lambda_0* requires compilation of additional C++ code performing the E(n) and E(r) calculations. To do this, browse to your **oSCR** folder and, at the command line/terminal, enter `R CMD SHLIB mysecrdesign.c`. To source the compiled code within R, use `dyn.load("oSCR/mysecrdesign.so")`, as shown in the examples available in the repository. An example is given in *example-spatial-lambda0.R*.

All other files reproduce the designs in our paper and results derived from these. Intermediate output objects used in the plots and tables in the paper are saved in the *output* folder.

- Efford, M., & Boulanger, J. (2019). Fast evaluation of study designs for spatially explicit capture-recapture. Methods in Ecology and Evolution, 10, 1529–1535.
- Sutherland, C., Royle, A., & Linden, D. (2018). oscr: Multi-session sex-structured spatial capture-recapture models [Computer software manual]. (R package version 0.42.0)
- Efford, M. G. (2019). secrdesign: Sampling Design for Spatially Explicit Capture-Recapture. R package version 2.5.7.
- Wolters, M.A (2015). A Genetic Algorithm for Selection of Fixed-Size Subsets with Application to Design Problems. Journal of Statistical Software, 68(1), 1-18.