Using spatial data in **secr** 4.5

Murray Efford

2022-03-04

Contents

Introduction

These notes explain how **secr** uses spatial data. Spatial data are used in **secr** to

1. locate detectors (`read.traps`, `read.capthist`)
2. map the extent of habitat near detectors (`make.mask`)
3. attach covariates to traps or mask objects (`addCovariates`)
4. delimit regions of interest (`region.N` and other functions)

Some spatial results may be exported, particularly the raster density surfaces generated by `predictDsurface` from a fitted model.

Internally, **secr** uses a very simple concept of space. The locations of detectors (traps), the potential locations of activity centres (habitat mask) and the simulated locations of individuals (popn) are described by Cartesian (x-y) coordinates assumed to be in metres. Distances are Euclidean unless specifically modelled as non-Euclidean (secr-noneuclidean.pdf). Only relative positions matter for the calculations, so the origin of the coordinates is arbitrary. The map projection ('coordinate reference system' or CRS) is not recorded.

Most spatial computations in **secr** (distances, areas, overlay etc.) use internal R and C++ code. Polygon and transect detectors are represented as dataframes in which each row gives the x- and y-coordinates of a vertex and topology is ignored (holes are not allowed).

The simple approach works fine within limits (discussed later), but issues arise when **secr**

- exchanges spatial data (regions, covariates or predicted density) with other software, or
- uses functions from R spatial packages, especially **sf** and **spsurvey**.

Spatial data in R

To use spatial data or functions from external sources in **secr** it helps to know a little about the expanding spatial ecosystem in R.

R packages for spatial data

Several widely used packages define classes and methods for spatial data ('Used by' in the following table is the number of CRAN packages from `crandep::get_dep` on 2022-02-03).

Package	Scope	Year	Used by	Citation	Relevant S4 classes
sp	vector	2005–	664	Pebesma & Bivand (2005)	SpatialPolygons, SpatialPolygonsDataFrame, SpatialGridDataFrame
raster	raster	2010–	507	Hijmans (2022b)	RasterLayer
sf	vector	2016–	434	Pebesma (2018)	sfg, sfc, sf
stars	both	2018–	39	Pebesma (2022)	stars
terra	both	2020–	37	Hijmans (2022a)	SpatVector, SpatRaster

The reader should already understand the distinction between vector and raster spatial data. There are many resources for learning about spatial analysis in R that may be found by web search on, for example 'R spatial data'. The introduction by Claudia Engels at https://cengel.github.io/R-spatial/ covers both **sp** and **sf**.

The capability of **sp** is being replaced by **sf** (https://r-spatial.github.io/sf/articles/) and **raster** is being replaced by **terra** (https://rspatial.org/terra/pkg/1-introduction.html). The more recent packages tend to be faster. **sf** implements the 'simple features' standard.

Geographic vs projected coordinates

QGIS has an excellent introduction to coordinate reference systems (CRS) for GIS. Coordinate reference systems may be specified in many ways; the most simple is the 4- or 5-digit EPSG code (search for EPSG on the web).

Geographic coordinates (EPSG 4326, ignoring some details) specify a location on the earth's surface by its latitude and longitude. This is the standard in Google Earth and GPS[1].

Spatial data in secr

Input of detector locations

secr uses relative Cartesian coordinates. Detector coordinates from GPS should therefore be projected from geographic coordinates before input to **secr**[2]. Most of the R spatial packages include projection functions. Here is a simple example using `st_transform` from the **sf** package:

```
library(sf)
# geographic coordinates (decimal degrees)
# longitude before latitude
df <- data.frame(x = c(174.9713, 174.9724, 174.9738), y = c(-41.3469, -41.3475, -41.3466))
```

[1] Geographic Positioning Systems.

[2] This does not apply if you are using an artificial coordinates (e.g., from `make.grid()`) rather than importing actual locations.

```
# construct sf object
latlon <- st_as_sf(df, coords = 1:2)
# specify initial CRS: WGS84 lat-lon
st_crs(latlon) <- 4326
st_coordinates(latlon)
```

```
##          X        Y
## 1 174.9713 -41.3469
## 2 174.9724 -41.3475
## 3 174.9738 -41.3466
```

```
# project to Cartesian coordinate system, units metres
# EPSG:27200 is the old (pre-2001) NZMG
trps <- st_transform(latlon, crs = 27200)
st_coordinates(trps)
```

```
##         X       Y
## 1 2674948 5982573
## 2 2675038 5982504
## 3 2675157 5982602
```

Adding spatial covariates to a traps or mask object

SECR models may include covariates for each detector (e.g., trap or searched polygon) in the detection model (parameters g_0, λ_0, σ etc.) and for each point on the discretized habitat mask in the density model (parameter D).

Covariates measured at detector locations may be included in the text files read by `read.traps` or `read.capthist`.

Covariates measured at each point on a habitat mask may be included in a file or data.frame input to `read.mask`, but this is an uncommon way to establish mask covariates. More commonly, a habitat mask is built using `make.mask` and initially has no covariates,

The function `addCovariates` is a convenient way to attach covariates to a traps or mask object *post hoc*. The function extracts covariate values from the 'spatialdata' argument by a spatial query for each point on a mask. Options are

spatialdata	Notes
character	name of ESRI shapefile, excluding '.shp'
sp::SpatialPolygonsDataFrame	
sp::SpatialGridDataFrame	
raster::RasterLayer	
secr::mask	covariates of nearest point
secr::traps	covariates of nearest point
terra::SpatRaster	new in 4.5.3
sf::sf	new in 4.5.3

Data sources should use the coordinate reference system of the target detectors and mask (see previous section).

Functions with 'poly' or 'region' spatial argument

Several **secr** functions use spatial data to define a region of interest (i.e. one or more polygons). From secr 4.5.3, all such polygons may be defined as

- 2-column matrix or data.frame of x- and y-coordinates
- SpatialPolygons or SpatialPolygonsDataFrame S4 classes from package **sp**
- SpatRaster S4 class from package **terra**
- sf or sfc S4 classes from package **sf** (POLYGON or MULTIPOLYGON geometries)

Data in these formats are converted to an object of class sfc by the documented internal function `boundarytoSF`. The S4 classes allow complex regions with multiple polygons (islands), possibly containing 'holes' (lakes).

This applies to the following functions and arguments:

secr function	Argument
buffer.contour	poly
deleteMaskPoints	poly
esa.plot	poly
make.mask	poly
make.systematic	region
mask.check	poly
pdot.contour	poly
PG	poly
pointsInPolygon	poly*
region.N	region*
sim.popn	poly
subset.popn	poly
trap.builder	region
trap.builder	exclude

* `pointsInPolygon` and `region.N` also accept a habitat mask.

GIS functionality imported from other R packages

Some specialised spatial operations are out-sourced by **secr**:

secr function	Operation	Other-package function	Reference
randomHabitat	simulated habitat	raster::adjacent raster::clump	Hijmans 2022b
nedist	non-euclidean distances	gdistance::transition gdistance::geoCorrection gdistance::costDistance	van Etten 2017
discretize	cell overlap with polygon(s)	sf::st_intersection sf::st_area	Pebesma 2018
polyarea	area of polygon(s)	sf::st_area	
make.mask	polybuffer mask type	sf::st_buffer	
rbind.capthist	merge polygon detectors	sf::st_union	
trap.builder	SRS sample	sf::st_sample	
trap.builder	GRTS sample (spsurvey >= 5.3.0)	spsurvey::grts	Dumelle et al. 2022

(Table updated for **secr** 4.5.4 on 2022-03-04)

Exporting raster data for use in other packages

A mask or predicted density surface (Dsurface) generated in **secr** may be used or plotted as a raster layer in another R package. **secr** provides `rast` and `raster` methods for **secr** mask and Dsurface objects, based on

the respective generic functions exported by **terra** and **raster**. These return SpatRaster and RasterLayer objects respectively. For example,

```
library(secr)
```

```
## This is secr 4.5.4 pre-release. For overview type ?secr
```

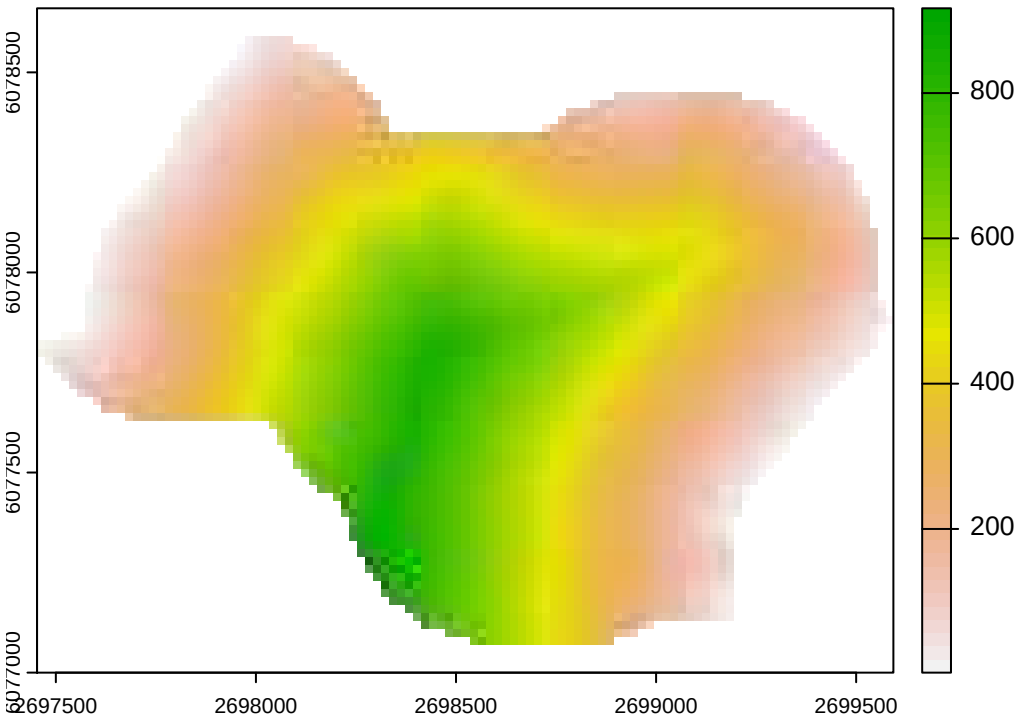
```
summary(possummask)
```

```
## Object class      mask
## Mask type         trapbuffer
## Number of points  5120
## Spacing m         20
## Cell area ha      0.04
## Total area ha     204.8
## x-range m         2697463 2699583
## y-range m         6077080 6078580
## Bounding box
##         x       y
## 1 2697453 6077070
## 2 2699593 6077070
## 4 2699593 6078590
## 3 2697453 6078590
##
## Summary of covariates
##    d.to.shore
##  Min.   :  2.236
##  1st Qu.:200.113
##  Median :370.804
##  Mean   :389.243
##  3rd Qu.:560.824
##  Max.   :916.692
```

```
# make SpatRaster object from mask covariate
r <- rast(possummask, covariate = 'd.to.shore')
print(r)
```

```
## class       : SpatRaster
## dimensions  : 76, 107, 1  (nrow, ncol, nlyr)
## resolution  : 20, 20  (x, y)
## extent      : 2697453, 2699593, 6077070, 6078590  (xmin, xmax, ymin, ymax)
## coord. ref. :
## source      : memory
## name        :      tmp
## min value   : 2.236068
## max value   : 916.6924
```

```
terra::plot(r)
```



Limits of the Cartesian model in secr

Distances computed in large studies

Distances on the curved surface of the earth are not well represented by Euclidean distances when the study area is very large, as happens with large carnivores such as grizzly bears and wolverines. This has led some authors to use more rigorous distance algorithms (reference?). This is not possible in **secr** because there is no record of the projected coordinate reference system used for the detectors and habitat mask.

References

Dumelle, M., Kincaid, T. M., Olsen, A. R., and Weber, M. H. (2022) spsurvey: Spatial Sampling Design and Analysis. R package version 5.3.0. https://CRAN.R-project.org/package=spsurvey

Hijmans, R. J. (2022a) terra: Spatial Data Analysis. R package version 1.5-14. https://rspatial.org/terra/

Hijmans, R. J. (2022b) raster: Geographic Data Analysis and Modeling. R package version 3.5-15. https://CRAN.R-project.org/package=raster

Pebesma, E. (2018) Simple features for R: standardized support for spatial vector data. *The R Journal* 10(1), 439–446. https://doi.org/10.32614/RJ-2018-009

Pebesma, E. (2022) stars: Spatiotemporal Arrays, Raster and Vector Data Cubes. https://r-spatial.github.io/stars/, https://github.com/r-spatial/stars/

Pebesma, E.J. and Bivand, R. S. (2005) Classes and methods for spatial data in R. *R News* 5(2), 9–13. https://cran.r-project.org/doc/Rnews/Rnews_2005-2.pdf

van Etten, J. (2017) R package gdistance: Distances and routes on geographical grids. *Journal of Statistical Software* 76(1), 1–21. https://doi.org/10.18637/jss.v076.i13