Setting up data for analysis with secr

University of St Andrews

Gather SECR data

SECR surveys use detectors at fixed locations to record the presence of individually identifiable animals at those locations. Detectors can be camera-traps, hair snares and dung surveys, live-captures, or acoustic detectors.

Set up data

The R package `secr` provides methods for estimating animal abundance from SECR data under many different conditions. This sheet summarizes getting your data into the format `secr` wants.

Analyse data

Once the data has been set up, use it to build SECR models and extract results on animal abundance, detectability, and important covariates.

(1) Make the detector file

Each line of `trapfile` contains the location of each detector (e.g. camera), plus any extra information about that detector.

TrapID	X	Y	Effort	/	tri	temp
A1	0	0	10 20	/	0.6	25
A2	5	0	10 19	/	0.9	23
A3	0	5	0 20	/	0.8	31

1 **TrapID**, **X**, and **Y** must be specified in the order given. **X** and **Y** contain the detector locations.

2 **Effort** records length of time each detector recorded for (optional). One value per occasion, separated by white space

3 Any other variables record covariates at the detectors (optional). These are stored to the right of the "/" column (also optional).

Detector covariates only used if detection function parameters vary across traps (`g0, lambda0, sigma`).
If using multiple sessions with detector changes between sessions, need one trapfile per session (see below).
Save as a .txt file to read into R later (.csv and .xlsx options also available).
Header row should begin with a # if saving as .txt

(2) Make the capture history file

Each line of `captfile` contains one detection, with ID variables recordings information about that detection.

Session	Animal	Occasion	TrapID
1	z001	1	A2
1	z174	2	A1
1	z024	1	A1

1 Each detection is recorded as a **session** identifier, **animal** identifier, **occasion** identifier.

2 Each detection includes a detector identifier, either as **trapID** (as above) or as X- and Y-coordinates (replace **trapID** with two columns **X**, and **Y)**

Session and occasion columns required even if you only use one session or occasion.
Occasion must be an integer starting from 1.
Save as a .txt file with header row starting with # (.csv and .xlsx options also available)

(3) Read it all in

Load both your `trapfile` and `captfile` files with `read.capthist`.

```
ch <- read.capthist(captfile = "ch.txt", trapfile = "tf.txt",
  detector = "count", fmt = "trapID", trapcovnames =
  c("tri", "temp), binary.usage = FALSE)
```

Important options

captfile, trapfile - the files made in the previous steps.

detector - specifies the type of detector you have. Most camera trap surveys will use "multi", "proximity" or "count".

fmt - if trapID used as detector identifier in `captfile` then `fmt = "trapID"`. If X and Y used then `fmt = "XY"`.

trapcovnames - names of covariates in `trapfile`

binary.usage - indicates if continuous effort variable present.

(4) Make the habitat mask

A mask is a **set of square grid cells** representing habitat **in the vicinity of detectors** that is **potentially occupied**.

A **mask object** is a 2-column dataframe, each row gives the **x- and y-coordinates** of the centre of one cell.

Constructing masks from detectors with `make.mask`

```
my_traps <- traps(ch)
my_mask <- make.mask(my_traps, buffer = 24000, spacing
  = 1000, type = "trapbuffer")
```

Makes a grid extending 24km N, S, E and W of any detectors

Puts mask points down at 1km intervals within the grid

Just 4 mask points for illustration

Make your own mask and `read.mask`

```
my_mask_df <- data.frame(X = c(0,1,0,1), Y = c(0,0,1,1),
  elevation = c(0,110,80,30))
my_mask <- read.mask(data = my_mask_df, spacing = 1)
```

Remember that `my_mask_df` must include the buffer region.

X	Y	elevation
0	0	0
1	0	110
0	1	80
1	1	30

Optional covariates

Sessions

A **session** is a sampling block that is treated as **independent.**

Can be spatial (arrays far apart enough that no animals are detected on both) or temporal.

```
ch <- read.capthist(
   captfile="ch.csv",     trapfile
   = c("sess1.csv",
   "sess2.csv")
my_mask <-
  make.mask(traps(ch))
```

One trap file per session. 1st file used for 1st session, 2nd file for 2nd session, etc.

Detector types

"multi" - animals can be detected at most once across all detectors in each occasion.

"proximity" - animals can be detected at most once at each detector in each occasion.

"count" - animals can be detected any number of times at each detector in each occasion.

See `?detector` for others.

(5) Add mask covariates

Mask covariates are used to model density (`D`), not detection parameters (`g0, lambda0, sigma`).

Adding covariates from a dataframe

```
covariates(my_mask) <-
  data.frame(elevation =
  c(0,110,80,30), temp =
  c(25,26,36,37)
```

can also add covariates before `read.mask` as in the bottom box in (4)

Adding covariates from a spatial data source

Assumes you have covariates stored in a spatial data source, which can be e.g. an ESRI polygon shapefile, SpatialPolygonsDataFrame, SpatialGridDataFrame (called `spdata` below)

```
addCovariates(object = ch,
  spatialdata = spdata, columns =
  c("elevation", "temp"))
```

Buffers

Choose buffer width large enough that animals beyond the buffer have **negligible chance** of being detected.

Rough rule of thumb is buffer > 4*sigma. Can get a rough estimate of sigma with `RPSV(ch,CC=TRUE)`.

Spacing

Too few grid points means a poor approximation of likelihoods, too many points slows down model fitting.

Rough rule of thumb is spacing < 1*sigma, and try for 1000-3000 grid points.

Analysing data with secr

Gather SECR data

SECR surveys use detectors at fixed locations to record the presence of individually identifiable animals at those locations. Detectors can be camera-traps, hair snares and dung surveys, live-captures, or acoustic detectors.

Set up data

The R package `secr` provides methods for estimating animal abundance from SECR data under many different conditions. First, you need to get your data into the format `secr` wants.

Analyse data

This sheet shows you how to build SECR models and extract results on animal abundance, detectability, and important covariates.

(1) Read in SECR inputs

To build models in secr you need to have already loaded:

1. A "capthist" object, which contains the capture histories and the trap locations
2. A "mask" object, a set of grid cells that defines the area that is potentially occupied and not so far from detector locations that observations are extremely unlikely.

```
ch <- read.capthist(captfile = "ch.txt", trapfile = "tf.txt",
   detector = "count", fmt = "trapID")
my_traps <- traps(ch)
my_mask <- make.mask(my_traps, buffer = 24000,
  spacing = 1000, type = "trapbuffer")
```

See the guide on "Setting up data" for more details

Detection models

A core SECR assumption is that detection probability (or frequency) **decreases with distance** to activity centre.

Shape is given by the **detection function** (`detectfn` in `secr.fit`), with a small number of parameters to be estimated.

`g0` detection models

These model the **probability** of detection. The most common option is "half-normal" (HN), with parameters `g0` and `sigma`, see `?detectfn` for others.

```
m0a <- secr.fit(ch, detectfn = "EX",    mask = my_mask,
   model = list(D ~ 1,     g0 ~ 1, sigma ~ 1))
```

"EX" function, another option

`lambda0` detection models

These model the **hazard** of detection. They are useful for quicker computation (especially for "count" detectors).

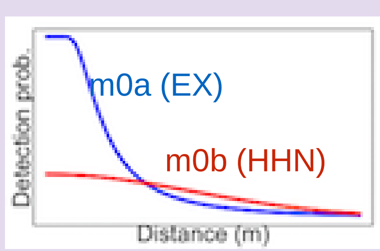
The most common option is "hazard half-normal" (HHN), with parameters `lambda0` and `sigma`, see `?detectfn` for others.

```
m0b <- secr.fit(ch, detectfn =
   "HHN",    mask = my_mask,
   model = list(D ~ 1, lambda0 ~
   1, sigma ~ 1))
```

`lambda0` and `g0` are mathematically equivalent and the choice between them is not crucial. Half-normal (HN or HHN) are good default options.

Plot detection functions

`plot(m0a)`



(2) Fit a model

SECR models jointly estimate two spatial models, one for animal density and one for the detection process.

Run SECR models with `secr.fit`, starting with the simplest possible model.

```
m0 <- secr.fit(ch, detectfn = "HHN",    mask = my_mask,
   model = list(D ~ 1, lambda0 ~ 1, sigma ~ 1))
```

- density, ~ 1 for constant density
- Detection function parameters, `lambda0` control encounter rate, `sigma` controls range of animal movement

! '~ 1' means no covariate effects, and a single parameter is estimated for each of `D, lambda0,` and `sigma`

Including covariates

Any of `D, lambda0,` and `sigma` can depend on covariates in the call to `secr.fit`.

```
m1 <- secr.fit(ch, detectfn = "HHN",    mask = my_mask,
   model = list(D ~ elev, lambda0 ~ water, sigma ~ 1))
```

- Encounter hazard `lambda0` depends on whether detector is close to water
- Density depends on elevation

Very flexible e.g. can do regression splines with `D ~ s(elev)`

! Covariates on density (`D`) must be attached to the mask object, covariates on detection parameters (`g0, lambda0, sigma`) must be attached to the trap object.

```
coef(m1)
```

```
Beta parameters (coefficients)
                 beta   SE.beta        lcl        ucl
D              -9.5184241 0.27550956 -10.0584129 -8.9784353
D.elev          0.2443394 0.39160813  -0.5231985  1.0118772
lambda0        -4.4403272 0.17332682  -4.7800415 -4.1006128
lambda0.WaterYes 0.2277942 0.27803197  -0.3171385  0.7727268
sigma           8.8583684 0.08326936   8.6951634  9.0215733
```

`secr` has a number of automatically generated "canned predictors" that can be referred to directly in formulae without needing to be constructed. These include `b` (learned animal responses to detectors), `k` (site learned response) and `session`, `t` and `T` (time effects), among others.

(3) Inspect model output

To view model output use `print(m0)`

```
N animals      :  14
N detections   :  99
N occasions    :  1
Count model    :  Poisson
Mask area      :  211725 ha

Model          :  D~1 lambda0~1 sigma~1
Fixed (real)   :  none
Detection fn   :  hazard halfnormal
Distribution   :  poisson
N parameters   :  3
Log likelihood :  -210.31
AIC            :  426.6199
AICc           :  429.0199

Beta parameters (coefficients)                          <- coef(m0)
           beta    SE.beta        lcl       ucl
D       -9.510329 0.26789360 -10.035391 -8.985267
lambda0 -4.387800 0.16494892  -4.711094 -4.064506
sigma    8.852195 0.08157649   8.692308  9.012082

Variance-covariance matrix of beta parameters           <- vcov(m0)
                   D      lambda0        sigma
D        0.0717669818 -0.0006783997 -0.0008909627
lambda0 -0.0006783997  0.0272081446 -0.0088657027
sigma   -0.0008909627 -0.0088657027  0.0066547240

Fitted (real) parameters evaluated at base levels of covariates   <- predict(m0)
       link     estimate  SE.estimate          lcl          ucl
D       log 7.408266e-05 2.020773e-05 4.382129e-05 1.252414e-04
lambda0 log 1.242805e-02 2.064016e-03 8.994936e-03 1.717147e-02
sigma   log 6.989717e+03 5.711466e+02 5.956917e+03 8.201582e+03
```

Main results are in this last table. Density is in animals per hectare.

Model selection

Model selection is by AIC or AICc (small sample size)

```
AIC(m0,m0a,m0b,m1)
```

Goodness-of-fit tests are underdeveloped but see `secr.test`.

Multi-session models

```
ch <- read.capthist(captfile="ch.csv",     trapfile =
   c("sess1.csv", "sess2.csv")
my_mask <- make.mask(traps(ch))
```

Can run `secr.fit` as in (2). Parameters are shared between sessions by default but any of `D, lambda0,` and `sigma` can be session-specific.

```
m2 <- secr.fit(ch, detectfn = "HHN", mask = my_mask, model
   = list(D ~ 1, lambda0 ~ 1, sigma ~ session))
```

Covariate effects can vary by session.

```
m3 <- secr.fit(ch, detectfn = "HHN", mask =
   my_mask, model=list(D ~ elev*session, lambda0 ~
   1, sigma ~ session))
```

secr version 4.1.0. Package created by Murray Efford. Learn more about the material shown here with the secr vignettes:secr-overview, secr-densitysurfaces, secr-multisession, secr-varyingeffort, available at https://www.otago.ac.nz/density/SECRinR.html.