Telemetry data in **secr 3.0**

Murray Efford

2017-04-10

Contents

Introduction

In some capture–recapture studies there are additional data from radiotelemetry of a sample of animals. Telemetry fixes provide an unbiased sample of animal activity, unlike detections at fixed points (traps, cameras or hair snares) or from searching circumscribed areas. Telemetry data therefore provide a direct estimate of the spatial scale of activity, which is represented in spatially explicit capture–recapture (SECR) by the parameter σ. Telemetry data also reduce uncertainty regarding the location of animals' centres relative to detectors, so detection histories of telemetered animals may improve estimates of other parameters.

This document explains and demonstrates the use of telemetry data in the R package **secr** 3.0, particularly the use of the 'telemetry' detector type. Many changes were made between **secr** 2.10 and **secr** 3.0; these notes, and the documentation for the various telemetry-related functions in **secr** 3.0, supercede any previous documentation.

A word of warning. Telemetry is used in **secr** to augment SECR analyses, particularly the estimation of population density. **secr** is not intended for the detailed analysis of telemetry data *per se*. There is no provision in the data structure for recording the time of each fix, except as each relates to a discrete sampling occasion. Nor is there provision for associating behavioural or environmental covariates with each fix.

Example

We start with a concrete example based on a simulated dataset. The code for the simulation is in Appendix 1.

The first step is to combine the capthist objects `trCH` (trapping data) and `teCH` (telemetry fixes).

```
library(secr)
```

```
combinedCH <- addTelemetry(trCH, teCH)
```

Generating plots is straightforward. By default, `plot.capthist` displays the captures and ignores telemetry fixes. The plot type "telemetry" displays the fixes and distinguishes those of animals that also appear in the (unplotted) capture data of the combined object.

```
par(mfrow = c(1,2), mar = c(2,2,3,2))
plot(traps(trCH), border = 150, bty = 'o') # base plot
plot(combinedCH, title = 'Trapping', tracks = TRUE, add = TRUE)
plot(traps(trCH), border = 150, bty = 'o') # base plot
plot(combinedCH, title = 'Telemetry', type = 'telemetry', tracks = TRUE, add = TRUE)
```

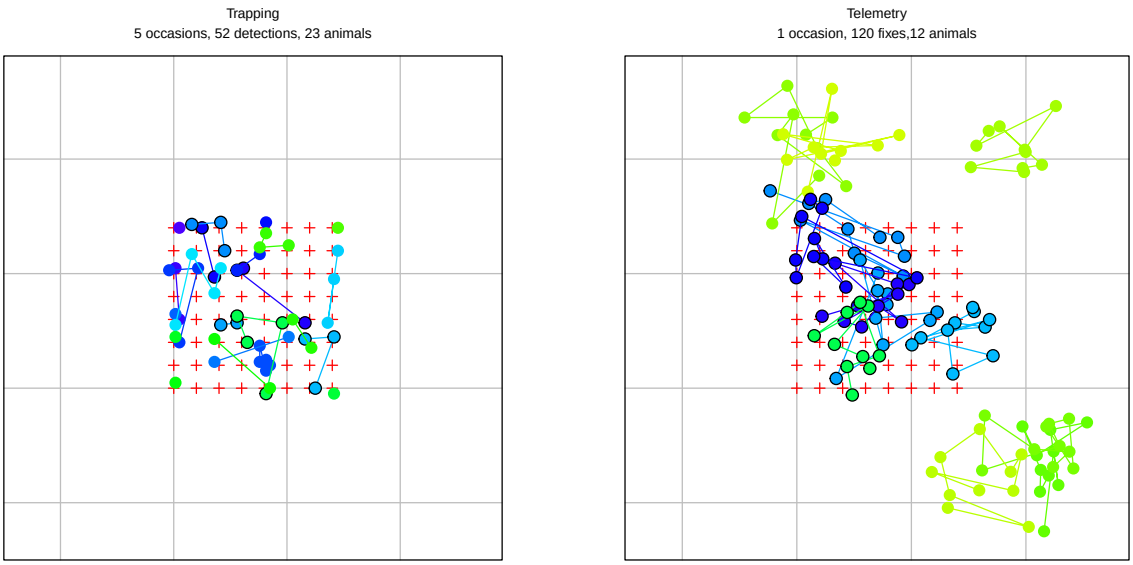

Fig. 1. Simulated trapping and telemetry data. The 5-day trapping study (left) yielded 29 recaptures. Telemetry data were obtained on 10 occasions for each of 12 animals (right). Points for animals that were both trapped and telemetered are ringed in black. Colours distinguish individuals.

Next we fit trapping-only and joint trapping-and-telemetry models:

```
mask <- make.mask(traps(trCH), buffer = 100, type = 'trapbuffer', nx = 32)
fit.tr <- secr.fit(trCH, mask = mask, detectfn = 'HHN', trace = FALSE)
fit.combined <- secr.fit(combinedCH, mask = mask, detectfn = 'HHN', trace = FALSE)
collate(tr = fit.tr, combined = fit.combined)[1,,,]
```

```
## , , D
##
##          estimate SE.estimate      lcl      ucl
## tr       5.905711    1.331210 3.817462 9.136285
## combined 5.669295    1.252941 3.695283 8.697819
##
## , , lambda0
##
##           estimate SE.estimate        lcl       ucl
## tr       0.1182759  0.02991088 0.07260431 0.1926770
## combined 0.1012350  0.02088806 0.06784654 0.1510546
##
## , , sigma
##
##          estimate SE.estimate      lcl      ucl
## tr       22.02743    2.403076 17.79823 27.26157
## combined 24.06819    1.304535 21.64416 26.76369
```

Here, telemetry data greatly improves the precision of the estimated scale of movement σ, but the effect on estimates of the other two parameters is small.

Standalone telemetry data

Telemetry data are stored in modified **secr** capthist objects. A capthist object may comprise telemetry fixes only ('standalone telemetry') or telemetry fixes in association with capture–recapture data (composite telemetry and capture–recapture). In this section I describe standalone telemetry data. A composite capthist is formed by combining a telemetry-only object and a standard capthist object with `addTelemetry`, as described in the next section.

The 'traps' object for telemetry data

Telemetry data differ from all other SECR data in that the detection process does not constrain where animals are detected (telemetry provides a spatially unbiased sample of each animal's activity). That is clearly not the case for area-search and point detectors, which inevitably constrain where animals are detected. Nevertheless, for compatibility with the rest of **secr**, we associate telemetry data with a notional detector located at a point[1]. The 'traps' object for telemetry data comprises the coordinates of this point plus the usual attributes of a 'traps' object in **secr** (detector type, usage, etc.). Remember that the point is only a 'notional' detector - it is never visited.

The function `make.telemetry` generates a suitable object:

```
te <- make.telemetry()
summary(te)
```

```
## Object class      traps
## Detector type     telemetry
## Telemetry type    independent
```

```
str(te)
```

```
## Classes 'traps' and 'data.frame':    1 obs. of  2 variables:
##  $ x: num 0
##  $ y: num 0
##  - attr(*, "detector")= chr "telemetry"
##  - attr(*, "telemetrytype")= chr "independent"
```

The attribute `telemetrytype` is always 'independent' for standalone telemetry data; other possible values for composite data are described later.

Detector type

Every 'traps' object has an associated detector type (attribute `detector`, commonly 'multi' or 'proximity'). From **secr** 3.0 onwards this may be a vector with a different value for each occasion. The detector type for telemetry data is 'telemetry'. In a standalone telemetry capthist, all elements of `detector` are 'telemetry'.

But where are the data?

As for any other detector type, the body of a telemetry capthist is a 3-D array whose elements are the number of detections for each combination of animal, occasion and detector. Coordinates are stored separately in the 'telemetryxy' attribute. Use one of these functions to reveal the telemetry component of a capthist object `CH`:

1. `str(CH)`
2. `summary(CH)`
3. `plot(CH, type = 'telemetry')`

[1] A completely different approach was used in versions before 3.0.

4. `telemetryxy(CH)`

The attribute 'telemetryxy' is a list with one component for each animal. The fixes of each animal are sorted in chronological order.

Data input

Data for a telemetry-only object should be read with function `read.telemetry`, a simplified version of `read.capthist`. The input format for telemetry fixes follows the 'XY' format for captures, with one line per fix (see secr-datainput.pdf).

The first few lines of a text file containing telemetry data collected on 5 occasions might look like this –

```
1  10 1  -83.3  -20.04
1  10 2  -57.91  -4.77
1  10 3 -112.96  -7.51
1  10 4  -77.71 -75.79
1  10 5  -85.81 -42.45
1 101 1  143.06 170.48
1 101 2   99.22 145.49
etc.
```

The first column is a session code, the next an animal identifier ('10', '101'), the third an occasion number (1..noccasions) and the last two are the x and y coordinates. GPS coordinates should be projected (i.e. not latitude and longitude), and in metres if possible.

A file named 'telemetrydemo.txt' may be read with

```
CHt <- read.telemetry(file = "telemetrydemo.txt")
```

```
## No errors found :-)
```

```
head(CHt)
```

```
## Session =  1
## , , 1
##
##    1 2 3 4 5
## 4  1 1 1 1 1
## 9  1 1 1 1 1
## 10 1 1 1 1 1
## 11 1 1 1 1 1
## 12 1 1 1 1 1
## 14 1 1 1 1 1
```

```
telemetryxy(CHt)[['10']]  # coordinates of first animal
```

```
##         x      y
## 1  -83.30 -20.04
## 2  -57.91  -4.77
## 3 -112.96  -7.51
## 4  -77.71 -75.79
## 5  -85.81 -42.45
```

Input may be from a text file (named in argument 'file') or dataframe (argument 'data').

The body of the resulting capthist object merely tallies the number of detections per animal per session and occasion. The fixes for one session are stored separately in an attribute that is a list of dataframes, one per animal. Use `telemetryxy(CHt)` to retrieve this list.

The summary of a telemetry-only capthist is quirky:

```
summary(CHt)
```

```
## Object class      capthist
## Detector type     telemetry (5)
## Telemetry type    independent
##
## Counts by occasion
##                     1  2  3  4  5 Total
## n                  79 79 79 79 79   395
## u                  79  0  0  0  0    79
## f                   0  0  0  0 79    79
## M(t+1)             79 79 79 79 79    79
## losses              0  0  0  0  0     0
## detections         79 79 79 79 79   395
## detectors visited   0  0  0  0  0     0
## detectors used      0  0  0  0  0     0
##
## Empty histories :  79
## 79 telemetered animals, 0 detected
## 5-5 locations per animal, mean =  5, sd = 0
```

Even though each fix is counted as a 'detection' in the body of the final capthist object, none of the telemetered animals is considered to have been 'detected' in a conventional SECR sense. The telemetry-only capthist object includes a trivial traps object with a single point. The telemetry type of the traps for a telemetry-only capthist defaults to 'independent'.

Combining telemetry and capture–recapture

For the purpose of density estimation and modelling, standalone telemetry data are added to an existing spatial capture–recapture (capthist) data object with the function `addTelemetry`. The relationship between the telemetry and capture–recapture samples is determined by the `type` argument (default 'concurrent'). We first explain the possible telemetry types.

Types of telemetry data

secr distinguishes three types of telemetry data – independent, dependent and concurrent – that differ in how they relate to other SECR samples (capture–recapture data). Each type corresponds to a particular probability model (Efford et al. in prep.).

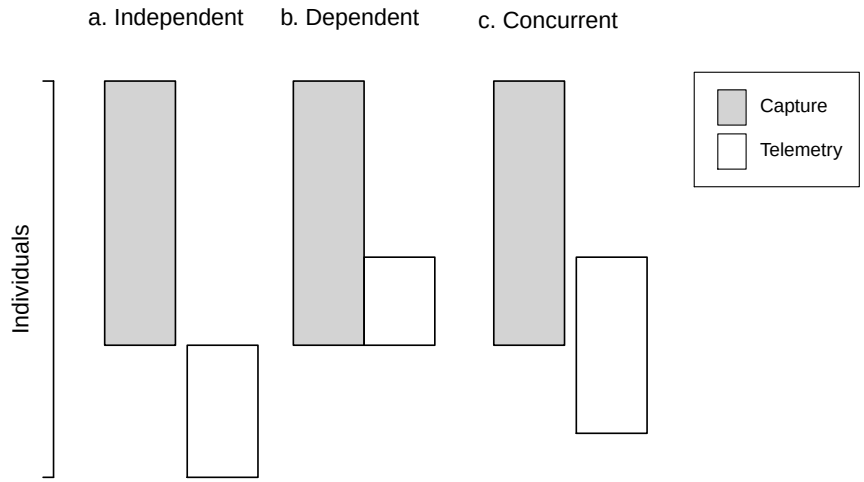

Fig. 2. Schematic relationship of capture–recapture data to three types of telemetry data. Vertical overlap indicates individuals that appear in both datasets.

1. Independent telemetry

Independent telemetry data have no particular relationship to spatial capture–recapture data except that they may be modelled using a shared value of the spatial-scale parameter σ, and possibly other spatial parameters. Telemetered animals do not have detection histories.

2. Dependent telemetry

Dependent telemetry data relate to a sample of animals detected during the capture–recapture study: an animal must be caught in that study to become telemetered, and no animal is telemetered and not otherwise detected (i.e. no detection history is all-zero).

3. Concurrent telemetry

Concurrent telemetry data are obtained for a sample of animals from the same regional population as the capture–recapture study. Telemetered animals appear stochastically in the capture–recapture sample with probability related to their location. Detection histories of some animals may be all-zero, and these are modelled. Whether the capture–recapture phase precedes or follows telemetry is not material.

What exactly does `addTelemetry` do?

The `addTelemetry` function forms a composite capthist object. Its usage follows -

```
addTelemetry (detectionCH, telemetryCH, type = c("concurrent", "dependent", "independent"),
    collapsetelemetry = TRUE, verify = TRUE)
```

Capture–recapture data in the argument 'detectionCH' form the basis for `addTelemetry`. The base capthist is modified in these ways –

- For all telemetry types `addTelemetry` extends the capture–recapture 'traps' object by adding a single (notional) detector location (duplicating the first).
- By default, the 'detector' attribute is extended by a single sampling occasion with type 'telemetry'; all telemetry data are associated with this occasion, regardless of how many occasions there were in the telemetry input. If `collapsetelemetry = FALSE` distinct telemetry occasions are retained.

- The 'usage' attribute is set to zero for the notional telemetry detector on capture occasions and for capture detectors on telemetry occasions. Other usage data from 'detectionCH' is retained.
- All-zero detection histories are generated for the 'concurrent' data type.
- The coordinates of telemetry fixes are transferred from `telemetryCH` as the attribute 'telemetryxy' of the output.
- If the data are independent then the labels of telemetered animals are prefixed by 'T' to reduce the chance of identity conflicts with animals in 'detectionCH'.
- By default, `addtelemetry` calls `verify.capthist` to check its output.

Composite data, different sessions

We use `addTelemetry` to combine telemetry data and capture–recapture data from the same session, or possibly for each of several sessions when the detectionCH and telemetryCH are parallel (equal-length) multi-session objects. It is also feasible to concatenate telemetry and capture–recapture data as separate sessions of a multi-session object with `MS.capthist`. The effect is similar to a single-session composite capthist with telemetrytype 'independent', because **secr** treats sessions as independent (i.e. individual histories do not span session boundaries). See the next section for an example.

Model fitting

Standalone telemetry data

We can estimate σ for a half-normal circular home-range model directly:

```
RPSV (CHt, CC = TRUE)
```

```
## [1] 24.86764
```

Note the CC argument (named for Calhoun and Casby 1958) that is required to scale the result correctly.

More laboriously:

```
fit0 <- secr.fit(CHt, buffer = 300, trace = FALSE)
```

```
## Warning in secr.fit(CHt, buffer = 300, trace = FALSE): detectfn not
## specified; using hazard half-normal (14)
```

```
predict(fit0)
```

```
##       link estimate SE.estimate      lcl     ucl
## sigma  log 24.86724   0.6995737 23.53347 26.2766
```

The detection function (argument `detectfn`) must be either hazard half-normal (14, 'HHN') or hazard exponential (16, 'HEX')[2]. The default detection function for a dataset with any telemetry component is 'HHN'. For telemetry-only data the likelihood is conditional on the number of observations, so the argument `CL` is set internally to TRUE. A large `buffer` value here brings $\hat{\sigma}$ from `secr.fit` closer to $\hat{\sigma}$ from `RPSV`. See below for more on the `buffer` argument.

See Technical notes for potential numerical problems.

[2]This constraint arises from the need internally to normalise the probability density function for each telemetry fix. The normalising constant for these functions is $1/(2\pi\sigma^2)$, whereas for most other possible values of `detectfn` it is hard to compute or the function does not correspond to a probability density.

Composite telemetry and capture–recapture data

Fitting a model to composite data should raise no further problems: `secr.fit` receives all the information it requires in the composite capthist input. The likelihood is a straightforward extension of the usual SECR likelihood, with some subtle differences in the case of dependent or concurrent telemetry (Efford et al. in prep.).

The use of detection functions expressed in terms of the hazard provides a more natural link between the model for the activity distribution and the model for detection probability. When a hazard function is used `secr.fit` automatically flips the default model for the first detection parameter from 'g0 ~ 1' to 'lambda0 ~ 1'.

Our introductory example fitted a model to single-session composite data. We can compare the results when the telemetry and trapping data are in separate sessions:

```
msCH <- MS.capthist(trCH, teCH)
fit.ms <- secr.fit(msCH, mask=mask, detectfn = 'HHN', trace = FALSE)
predict(fit.ms)
```

```
## $`session = Trapping`
##         link    estimate SE.estimate        lcl        ucl
## D        log  5.37684647  1.18964239  3.5030195  8.2530165
## lambda0  log  0.09257824  0.01869695  0.0625629  0.1369938
## sigma    log 26.01923992  1.15710224 23.8484137 28.3876678
##
## $`session = Telemetry`
##       link estimate SE.estimate      lcl      ucl
## sigma  log 26.01924    1.157102 23.84841 28.38767
```

Note: If the order of `teCH` and `trCH` had been reversed in `msCH` we would need to use details=list(autoini=2) to base parameter starting values on the trapping data, or provide start values manually.

Habitat mask for telemetry data

The centres of both detected and telemetry-only animals are assumed to lie on the habitat mask. Ensure the mask is large enough to encompass telemetry-only animals. A conservative approach is to buffer around the individual telemetry centroids. Using `teCH` from before:

```
centroids <- data.frame(t(sapply(telemetryxy(teCH), apply, 2, mean)))
mask1 <- make.mask(centroids, buffer = 100, type = 'trapbuffer')
```

For composite telemetry and capture–recapture, buffering should include the detector sites:

```
tmpxy <- rbind(centroids, data.frame(traps(trCH)))
mask2 <- make.mask(tmpxy, buffer = 100, type = 'trapbuffer')
```

```
par(mfrow = c(1,2))
plot(mask1)
plot(teCH, add = TRUE, title = 'Telemetry only')
plot(mask2)
plot(traps(trCH), add = TRUE)
plot(teCH, add = TRUE, title = 'Telemetry and detector sites')
```

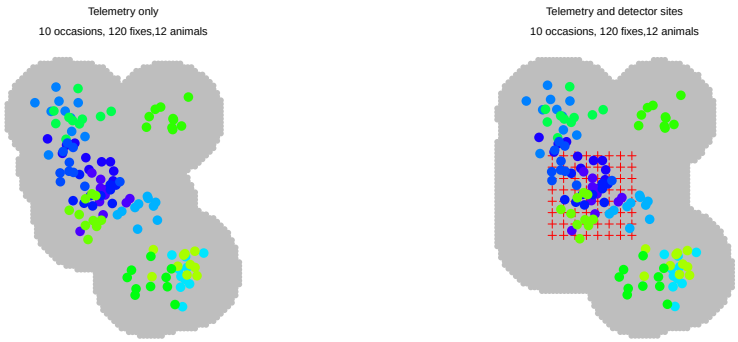

Fig. 3. Habitat masks prepared by buffering around telemetry sites (left) or both telemetry and detector sites (right).

From **secr** 3.0 on, the mask generated automatically by `secr.fit` buffers around both detector sites and telemetry fixes, as shown here. The 'buffer' argument in `make.mask` can be problematic when used with a standalone telemetry traps object because the notional detector location is an arbitrary point - it is better to use the centroid coordinates as input.

Simulation

Simulation of joint capture–recapture and telemetry data is a 2-step operation in **secr**, with the steps depending on the type of telemetry sampling. Here is an example of each type.

We choose to fix the number of observations per animal at 25 using the `exactN` argument of `sim.capthist`. The same effect can be achieved by increasing the number of occasions to 25 and setting `exactN = 1`.

Independent telemetry

For independent data there is no specified connection between the populations sampled, so we separately generate telemetry and capture–recapture datasets and stick them together.

```
# detectors
te <- make.telemetry()
tr <- make.grid(nx = 8, ny = 8, detector = "proximity")
pop1 <- sim.popn(tr, D = 10, buffer = 200)
pop2 <- sim.popn(core = tr, buffer = 200, Nbuffer = 20, Ndist = 'fixed')
trCH <- sim.capthist(tr,  popn = pop1, detectfn = "HHN", detectpar =
                         list(lambda0 = 0.1, sigma = 25))
teCH <- sim.capthist(te,  popn = pop2, detectfn = "HHN", detectpar =
                         list(sigma = 25), noccasions = 1, exactN = 25)
CHI <- addTelemetry(trCH, teCH, type = 'independent')
```

```
## No errors found :-)
```

```
session(CHI) <- 'Independent'
summary(CHI)
```

```
## Object class       capthist
## Detector type      proximity (5), telemetry
```

```
## Telemetry type    independent
## Detector number   64
## Average spacing   20 m
## x-range           0 140 m
## y-range           0 140 m
##
##  Usage range by occasion
##     1 2 3 4 5 6
## min 0 0 0 0 0 0
## max 1 1 1 1 1 1
##
## Counts by occasion
##                     1  2  3  4  5   6 Total
## n                  18 15 17 18 23  20   111
## u                  18  5  9  1  3  20    56
## f                  29  7 13  6  1   0    56
## M(t+1)             18 23 32 33 36  56    56
## losses              0  0  0  0  0   0     0
## detections         28 18 25 27 35 500   633
## detectors visited 24 16 22 19 26   0   107
## detectors used    64 64 64 64 64   0   320
##
## Empty histories :  20
## 20 telemetered animals, 0 detected
## 25-25 locations per animal, mean =  25, sd = 0
```

Dependent telemetry

For dependent data the telemetry sample is drawn from animals caught during the capture–recapture phase. This example uses the previously constructed 'traps' objects (`tr` and `te`). The original numbering of animals must be conserved (`renumber = FALSE`).

```
pop3 <- sim.popn(tr, D = 10, buffer = 200)
trCH <- sim.capthist(tr,  popn = pop3, detectfn = "HHN",
                     detectpar = list(lambda0 = 0.1, sigma = 25),
                     renumber = FALSE, savepopn = TRUE)

## select trapped animals from saved popn
pop3D <- subset(attr(trCH, 'popn'), rownames(trCH))
## sample 12 detected animals for telemetry
pop3Dt <- subset(pop3D, sample.int(nrow(pop3D), 12))
## simulate telemetry
teCHD <- sim.capthist(te, popn = pop3Dt, renumber = FALSE, detectfn = "HHN",
                     detectpar = list(sigma = 25), noccasions = 1, exactN = 25)
CHD <- addTelemetry(trCH, teCHD, type = 'dependent')
```

```
## No errors found :-)
```

```
session(CHD) <- 'Dependent'
summary(CHD)
```

```
## Object class      capthist
## Detector type     proximity (5), telemetry
## Telemetry type    dependent
## Detector number   64
```

```
## Average spacing   20 m
## x-range           0 140 m
## y-range           0 140 m
##
##  Usage range by occasion
##     1 2 3 4 5 6
## min 0 0 0 0 0 0
## max 1 1 1 1 1 1
##
## Counts by occasion
##                     1  2  3  4  5   6 Total
## n                  20 17 23 20 23  12   115
## u                  20  9  9  8  5   0    51
## f                  15 21  7  4  3   1    51
## M(t+1)             20 29 38 46 51  51    51
## losses              0  0  0  0  0   0     0
## detections         26 23 29 29 29 300   436
## detectors visited 20 18 24 25 24   0   111
## detectors used    64 64 64 64 64   0   320
## 12 telemetered animals, 12 detected
## 25-25 locations per animal, mean =  25, sd = 0
```

Concurrent telemetry

For concurrent telemetry a sample of animals is taken from the regional population without reference to whether or not each animal was detected in the capture–recapture phase. The original numbering of animals must be conserved (`renumber = FALSE`), as for dependent telemetry.

```
pop4 <- sim.popn(tr, D = 10, buffer = 200)
# select 15 individuals at random from larger population
pop4C <- subset(pop4, sample.int(nrow(pop4), 15))
# the original animalID (renumber = FALSE) are needed for matching
trCH <- sim.capthist(tr,  popn = pop4, renumber = FALSE, detectfn = "HHN",
                     detectpar = list(lambda0 = 0.1, sigma = 25))
teCHC <- sim.capthist(te, popn = pop4C, renumber = FALSE, detectfn = "HHN",
                     detectpar = list(sigma = 25), noccasions = 1, exactN = 25)
CHC <- addTelemetry(trCH, teCHC, type = 'concurrent')
```

```
## No errors found :-)
```

```
session(CHC) <- 'Concurrent'
summary(CHC)
```

```
## Object class      capthist
## Detector type     proximity (5), telemetry
## Telemetry type    concurrent
## Detector number   64
## Average spacing   20 m
## x-range           0 140 m
## y-range           0 140 m
##
##  Usage range by occasion
##     1 2 3 4 5 6
## min 0 0 0 0 0 0
## max 1 1 1 1 1 1
```

```
##
## Counts by occasion
##                      1  2  3  4  5   6 Total
## n                   17 16 20 19 16  15   103
## u                   17 10  7  6  2  14    56
## f                   29 13  9  4  1   0    56
## M(t+1)              17 27 34 40 42  56    56
## losses               0  0  0  0  0   0     0
## detections          23 27 24 22 23 375   494
## detectors visited   17 25 21 21 19   0   103
## detectors used      64 64 64 64 64   0   320
##
## Empty histories :  14
## 15 telemetered animals, 1 detected
## 25-25 locations per animal, mean =  25, sd = 0
```

Plotting to compare simulated data

```
par(mfrow = c(1,3), mar = c(2,2,4,2), xpd = TRUE)
plot(traps(CHI), border = 200, gridlines = FALSE, bty = 'o')
plot(CHI, type = 'telemetry', tracks = TRUE, add = TRUE)

plot(traps(CHD), border = 200, gridlines = FALSE, bty = 'o')
plot(CHD, type = 'telemetry', tracks = TRUE, add = TRUE)

plot(traps(CHC), border = 200, gridlines = FALSE, bty = 'o')
plot(CHC, type = 'telemetry', tracks = TRUE, add = TRUE)
```

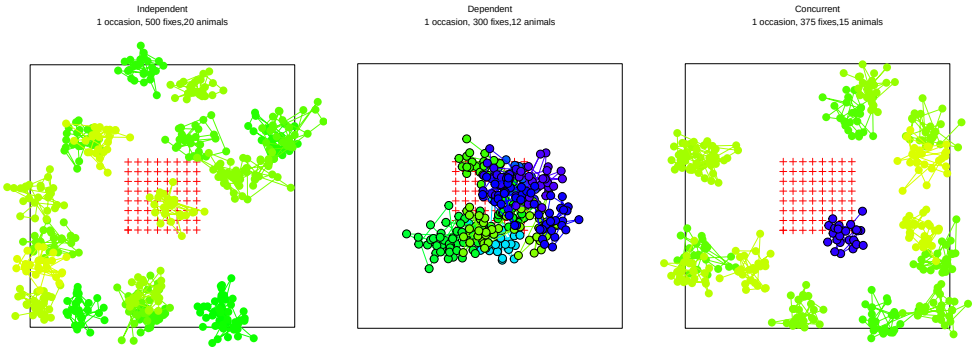

Fig. 3. Simulated telemetry data in relation to a capture–recapture grid (red crosses). Independently telemetered individuals are not recognised if they are caught on the grid. Dependent telemetry is restricted to animals caught on the grid. Individuals telemetered concurrently may or may not be caught, but are recognised when they are.

Technical notes

Assumption of common σ

Joint analysis of telemetry and capture–recapture data usually relies on the assumption that the same value of the parameter σ applies in both sampling processes. This does not hold when

- telemetry fixes have large measurement error that inflates σ, or
- the tendency of an animal to interact with a detector after encountering it varies systematically with distance from the home-range centre, or
- activity is not stationary and the telemetry and capture–recapture data relate to different time intervals.

The assumption may be avoided altogether by modelling distinct values of σ on trapping and telemetry occasions. This is readily achieved using the automatic predictor `tt` in the formula for sigma, as in `secr.fit(CH, detectfn = 'HHN', model = sigma~tt, ...)`. The model then has one level of sigma for non-telemetry occasions (tt = 'nontelemetry') and another for telemetry occasions (tt = 'telemetry'). However, this sacrifices much of the benefit from a joint analysis when the telemetry data are dependent or concurrent, and all benefit for independent telemetry data.

Numerical problems

Fitting joint telemetry and SECR models can be difficult - the usual computations in **secr** may fail to return a likelihood.

The problem may be partially diagnosed by inspecting the successive likelihoods computed by `secr.fit` when called with `trace = TRUE` or `details = list(debug = 1)`.

If the likelihood returned by `secr.fit` is zero (0) then the problem is almost certainly due to a near-zero value in a component of the telemetry likelihood. This occurs particularly in large datasets. The problem may be fixed by scaling the offending values by an arbitrary large number given in the details argument 'telemetryscale'. The required magnitude for 'telemetryscale' may be found by experimentation (try 1e3, 1e6, 1e9, 1e12 etc.). This ad hoc solution must be applied consistently if models are to be compared by AIC.

```
te <- make.telemetry()
teCH2 <- sim.capthist(te, popn = list(D = 2, buffer = 200), detectfn = "HHN", exactN = 100,
                      detectpar = list(sigma = 25), noccasions = 1)
mask <- make.mask(traps(teCH2), buffer = 200, type = 'trapbuffer', nx = 32)
# fails
fit1 <- secr.fit(teCH2, mask = mask, detectfn = 'HHN', CL = TRUE,
                 details = list(telemetryscale = 1))
# succeeds
fit1000 <- secr.fit(teCH2, mask = mask, detectfn = 'HHN', CL = TRUE,
                 details = list(telemetryscale = 1e3))
```

If the likelihood returned by `secr.fit` is NA rather than zero then the problem may be inappropriate starting values, poor model specification, or an unknown bug. It may help to use the longer-tailed detection function 'HEX' instead of 'HHN'.

Learned response

Learned responses (b, bk) are not expected in telemetry data. However, they may make sense for the 'SECR' occasions of a composite dataset (combined stationary detectors and telemetry). There is no way to avoid a global learned response (b) from propagating to the telemetry occasions (i.e. modelling different telemetry sigmas for animals detected or not detected in the pre-telemetry phase). A site-specific learned response, however, cannot propagate to the telemetry phase if there is a single telemetry 'occasion' because (i) no animal is detected at the notional telemetry detector in the pre-telemetry phase, and (ii) there is no opportunity for learning within the telemetry phase if all detections are on one occasion.

Limitations

In **secr** 3.0 some important functions have yet to be updated to work with telemetry data. These are listed in Appendix 2. Other limitations are described here.

Incompatible with area search

Telemetry data may not be combined with area-search (polygon) data except as independent data in distinct sessions. This is because the polygon data types presently implemented in **secr** must be constant across a session.

If the 'independent data, distinct-session' solution is inadequate you might try rasterizing the search area (function `discretize`).

Incompatible with mark–resight

Telemetry data may not be combined with mark-resight in **secr** 3.0 except possibly in distinct sessions (this has not been tested).

Incompatible with hybrid heterogeneity model

The `secr.fit` code for hybrid heterogeneity models (hcov) has yet to be updated.

Non-Euclidean distance

Non-Euclidean distance methods cannot be used with telemetry data at present (a very large distance matrix would be required).

References

Borchers, D. L. and Efford, M. G. (2008) Spatially explicit maximum likelihood methods for capture–recapture studies. *Biometrics* **64**, 377–385.

Calhoun, J. B. and Casby, J. U. (1958) Calculation of home range and density of small mammals. Public Health Monograph. No. 55. U.S. Government Printing Office.

Efford, M. G. (2011) Estimation of population density by spatially explicit capture–recapture analysis of data from area searches. *Ecology* **92**, 2202–2207.

Efford, M. G., Boulanger, J., Spencer, N. J., Warburton, B. and Stenhouse, G. In prep. Joint analysis of telemetry and capture–recapture data.

Gopalaswamy, A. M., Royle, J. A., Delampady, M., Nichols, J. D., Karanth, K. U. and MacDonald, D. W. (2012) Density estimation in tiger populations: combining information for strong inference. *Ecology* **93**, 1741–1751.

Sollmann, R., Gardner, B., Parsons, A. W., Stocking, J. J., McClintock, B. T., Simons, T. R., Pollock, K. H., and O'Connell, A. F. (2013). A spatial mark-resight model augmented with telemetry data. *Ecology* **94**, 553–559.

Whittington, J., Hebblewhite, M. and Chandler, R. (2016) Generalized spatial mark–resight models with an application to grizzly bears. *Journal of Applied Ecology* In press.

Appendix 1.

Code used to simulate data for initial demonstration

```
# simulate capthist objects (trCH, teCH)
library(secr)
# detectors
te <- make.telemetry()
tr <- make.grid(detector = "multi", nx = 8, ny = 8)

pop4 <- sim.popn(tr, D = 5, buffer = 100, seed = 567)
# select 12 individuals at random from larger population
pop4C <- subset(pop4, sample.int(nrow(pop4), 12))
# the original animalID (renumber = FALSE) are needed for matching
trCH <- sim.capthist(tr,  popn = pop4, renumber = FALSE, detectfn = "HHN",
                     detectpar = list(lambda0 = 0.1, sigma = 25), seed = 123)
session(trCH) <- 'Trapping'
teCH <- sim.capthist(te, popn = pop4C, renumber = FALSE, detectfn = "HHN",
                     detectpar = list(lambda0 = 1, sigma = 25), noccasions = 10, seed = 345)
session(teCH) <- 'Telemetry'
```

Appendix 2.

Functions for telemetry data

Function	Purpose
`addTelemetry`	combine capture-recapture and telemetry data in new capthist
`make.telemetry`	build a traps object for standalone telemetry data
`read.telemetry`	input telemetry fixes from text file or dataframe
`telemetered`	determine which animals in a capthist object have telemetry data
`telemetrytype`	extract or replace the 'telemetrytype' attribute of a traps object
`telemetryxy`	extract or replace telemetry coordinates from capthist
`xy2CH`	make a standalone telemetry capthist from a composite capthist

Telemetry-ready general functions

Function	Purpose
`derived`	Horvitz-Thompson-like density estimate
`join`	combine sessions of multi-session capthist object
`make.capthist`	build capthist object
`moves`	sequential movements*
`MS.capthist`	form multi-session capthist from separate sessions
`plot.capthist`	plotting (type = 'telemetry')
`rbind.capthist`	concatenate rows of capthist
`RPSV`, `MMDM`, `ARL`, `dbar`	indices of home-range size*
`secr.fit`	model fitting
`sim.capthist`	generate capthist data
`subset.capthist`	select subset of animals, occasions or detectors

Function	Purpose
`summary.capthist`	summary
`verify.capthist`	perform integrity checks
`verify.traps`	perform integrity checks

* these functions use the telemetry coordinates if the capthist is telemetry-only, otherwise the detection sites

General functions not ready for telemetry

Function	Purpose
`reduce.capthist`	change detector type or collapse occasions
`sim.secr`	parametric bootstrap fitted model
`simulate`	simulate from fitted model
`secr.test`	another parametric boostrap
`fx.total`	
`fxi.contour`	