secrlinear - spatially explicit capture–recapture for linear habitats

Murray Efford

2021-05-04

Contents

Introductory example **2**

Linear habitat masks **3**
Input from a polyline shapefile 4
Input from a SpatialLinesDataFrame 5
Input from a dataframe of coordinates 5
Input from a text file of coordinates 5

Network distance **5**

Detector layouts **7**

Simulating detection data **8**

Model fitting **9**

Advanced topics **10**
Population size and effective sampling area 10
Linear habitat covariates 11
Discontinuities and traversable non-habitat 12
Linear home-range size 13
Evaluating study designs 13

Limitations, tips and troubleshooting **16**

References **17**

Appendix. Extended water vole example. **17**

The R package **secrlinear** extends **secr** for animal populations in linear habitats. A habitat is considered to be 'linear' if it is natural to express population density as number per unit length (e.g., animals per km) rather than number per unit area (e.g., animals per ha). For example, species such as trout and riverine otters may live almost entirely in stream channels. **secrlinear** provides functions to manipulate linear habitat masks (the 'linearmask' class) and to approximate distances within a network defined by a linear mask. The linear mask and network distance function may be used directly in **secr** functions such as `secr.fit`. Minor changes to **secr** in version 2.9.0 provided for the analysis of linear populations (e.g., `sim.popn` now has a 'linear' option for its argument 'model2D').

Introductory example

Water voles (*Arvicola amphibius*) were trapped monthly in 1984 along 0.9 km of the River Glyme near Woodstock in Oxfordshire, U.K. (Efford 1985). Two sheet-aluminium traps were set at stations 20 m apart along one bank and checked morning and evening for 3 days. Voles were marked with individually colour-coded ear tags. We use data from June 1984. This was early in the vole breeding season when most voles were overwintered adults: only 3 were young-of-the-year and these were omitted.

Raw data files "Jun84capt.txt" and "glymetrap.txt" are provided in the 'extdata' folder of the **secrlinear** installation. These are in the format used by **secr** (see the vignette secr-datainput.pdf for details). We first load all libraries that will be needed for this vignette and import the capture data, exactly as for 2-dimensional habitat in **secr**. Although single-catch traps were used, we set the detector type to "multi" to avoid later warnings.

```r
library(secrlinear)    # also loads secr
library(rgdal)         # to read shapefiles
library(igraph)        # to investigate edge lengths
options(digits = 4)    # for more readable output
inputdir <- system.file("extdata", package = "secrlinear")
```

```r
captfile <- paste0(inputdir, "/Jun84capt.txt")
trapfile <- paste0(inputdir, "/glymetrap.txt")
arvicola <- read.capthist(captfile, trapfile, covname = "sex")
```

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

Next we need to define the linear habitat geometry. For one simple line (no branches) we can read the coordinates of the river bank from a text file. Coordinates are in a Cartesian system with an arbitrary origin; distances are in metres. We form a 'linearmask' object by cutting the line at 4-m intervals.

```r
habitatmap <-  paste0(inputdir, "/glymemap.txt")
glymemask <- read.linearmask(file = habitatmap, spacing = 4)
```

Now we can display the data:

```r
par(mar = c(1,1,4,1))
plot(glymemask)
plot(arvicola, add = TRUE, tracks = TRUE)
plot(traps(arvicola), add = TRUE)
```

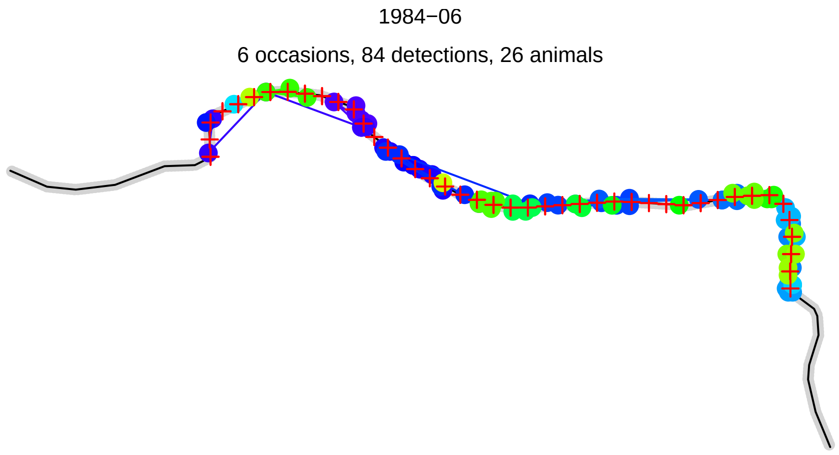

Fig. 1. Water vole captures along R. Glyme in June 1984. Traps (red crosses) spanned 860 m of river bank.

We can fit a spatially explicit model to the water vole data in three ways: (i) ignoring the linearity of the habitat (fit2DEuc), (ii) with a linear habitat map and the default Euclidean distance model (fit1DEuc), or

(iii) with both linear habitat and an appropriate non-Euclidean distance function (fit1DNet):

```
# 2-D habitat, Euclidean distance
fit2DEuc <- secr.fit(arvicola, buffer = 200, trace = FALSE)

# 1-D habitat, Euclidean distance
fit1DEuc <- secr.fit(arvicola, mask = glymemask, trace = FALSE)

# 1-D habitat, river distance
fit1DNet <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                details = list(userdist = networkdistance))
```

The second fit displays the warning "using Euclidean distances with linear mask". We compare the parameter estimates from the three models:

```
predict(fit2DEuc)
```

```
##        link estimate SE.estimate     lcl     ucl
## D       log   1.2798      0.3046  0.8079  2.0275
## g0    logit   0.2634      0.0978  0.1175  0.4899
## sigma   log  41.9103      4.1392 34.5507 50.8375
```

```
predict(fit1DEuc)
```

```
##        link estimate SE.estimate      lcl     ucl
## D       log 26.20059      5.2072 17.81459 38.5342
## g0    logit  0.07629      0.0111  0.05719  0.1011
## sigma   log 44.41595      4.3780 36.63023 53.8565
```

```
predict(fit1DNet)
```

```
##        link estimate SE.estimate      lcl      ucl
## D       log 26.52514     5.27114 18.03598 39.00995
## g0    logit  0.07283     0.01052  0.05472  0.09632
## sigma   log 47.39258     4.86151 38.78135 57.91590
```

Analysis with a 2-dimensional mask and Euclidean distances gives an estimated density of 1.3 voles per ha. When the mask is linear, density is expressed in animals per km (here $\hat{D} = 26.5$/km, SE 5.3/km). Using the correct distances (i.e. distances measured along the river with function `networkdistance`) has only a slight effect in this instance because the river is unbranched and nearly straight.

In the rest of this vignette we explore in more detail the options for linear SECR that are available in **secrlinear**.

Linear habitat masks

The representation of linear habitat in **secrlinear** is based on the SpatialLinesDataFrame object class defined in the package **sp** (Pebesma and Bivand 2005). This provides the habitat template - e.g., a map of a stream channel network. For ease of modelling the mask is discretized into equal-length line segments, each represented by the coordinates of its central point as for other habitat masks in **secr**. The discretized form is the most visible aspect of a linear mask. The underlying linear data are stored as an attribute ('SLDF'). The topology of the network is represented by an **igraph** (Csardi and Nepusz 2006) graph object saved as the attribute 'graph'. This is an undirected graph with the length of each edge stored as the attribute "weight".

Other objects (detectors, animals) are assumed to lie on the linear mask. In effect, their x-y locations are snapped to the nearest mask point (the discretized representation).

The function `read.linearmask` creates a linear mask from inputs that define the linear network. These may be a polyline shapefile (ESRI 1998), a SpatialLinesDataFrame, a dataframe of coordinates for line vertices, or a file containing coordinate data (as in the introductory example). The first two methods allow for complex combinations of lines, including branching networks. The 'spacing' argument of `read.linearmask` determines the coarseness of the discretization.

Input from a polyline shapefile

The 'file' argument of `read.linearmask` may be the name of a shapefile, including the '.shp' extension. The function `readOGR` from **rgdal** (Bivand et al. 2005) is used internally to create a SpatialLinesDataFrame from the shapefile.

We can try this with a shapefile for the Silverstream catchment near Dunedin, New Zealand, based on data from Land Information New Zealand (see ?Silverstream for details). The relevant files are 'silverstream.shp', 'silverstream.dbf' and 'silverstream.shx' in the 'extdata' folder of **secrlinear**.

```
habitatmap <- paste0(inputdir, "/silverstream.shp")
silverstreammask <- read.linearmask(file = habitatmap, spacing = 50)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\Murray\AppData\Local\Temp\RtmpGaiEAN\Rinst1a9012b55f8c\secrlinear\extdata\silverst
## with 82 features
## It has 2 fields
```

```
par(mar = c(1,1,1,1))
plot(silverstreammask)
```

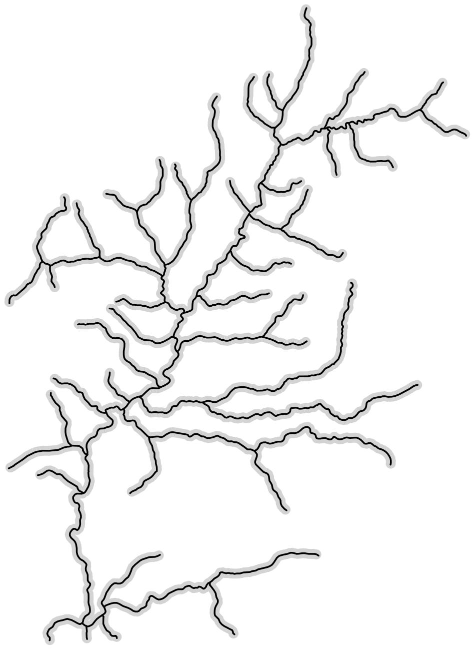

Fig. 2. Linear mask for Silverstream catchment, Dunedin, New Zealand. The river flows to the bottom left. Mask points are grey dots at 50-m spacing; these merge together in the plot.

The discretized length of a complex mask will tend to be shorter than the original linear network as incomplete segments are dropped from the ends of branches. See the following section for more details.

```
networklength <- sum(SpatialLinesLengths(attr(silverstreammask, "SLDF"))) / 1000
discrepancy <- networklength - masklength(silverstreammask)  # km
```

In this case the discrepancy is 0.06 km.

Input from a SpatialLinesDataFrame

The ‘data’ argument of `read.linearmask` may be a SpatialLinesDataFrame object. This allows for data to be imported from GIS sources other than shapefiles, and for selection of features, re-projection, or other manipulation.

```
habitatmap <- paste0(inputdir, "/silverstream.shp")
silverstreamSLDF <- rgdal::readOGR(dsn = habitatmap, layer = "silverstream")
silverstreammask <- read.linearmask(data = silverstreamSLDF, spacing = 50)
```

Input from a dataframe of coordinates

Alternatively, a simple mask may be constructed from a dataframe of coordinates. This example generates an artificial geometry.

```
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
linmask <- read.linearmask(data = xy, spacing = 20)
```

```
plot(linmask)
```

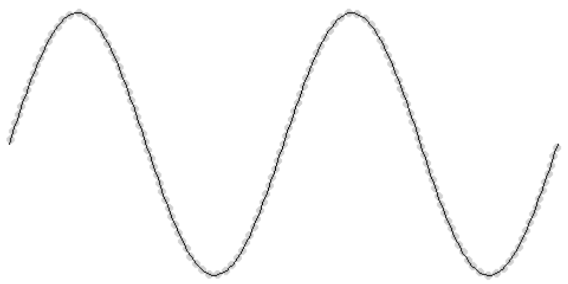

Figure 1: linearmask

Fig. 3. Artificial linear habitat mask. Each mask segment is 20 m long (pixel centres are indicated by a grey dot).

Input from a text file of coordinates

See water vole example.

Network distance

If the movement of animals is largely confined to the linear network then it makes sense to measure distances along the network rather than as the crow flies. Network distances are non-Euclidean. If the animals concentrate their movements around linear features, but do not use them exclusively, then other non-Euclidean approaches may be appropriate (cf Royle et al. 2013, Sutherland et al. 2015). These require the estimation of an additional parameter and use a 2-dimensional habitat mask.

The **igraph** function `shortest.distance` is used within the **secrlinear** function `networkdistance` to compute approximate distances on the network implied by a linear mask. In the case of a single line, the distance is simply the product of the mask spacing and the number of steps (intervening points + 1).

The interactive `showpath` function allows you to verify the distance computation for chosen points on the network:

```
# start interactive session and click on two points
showpath(silverstreammask, lwd = 3)
```

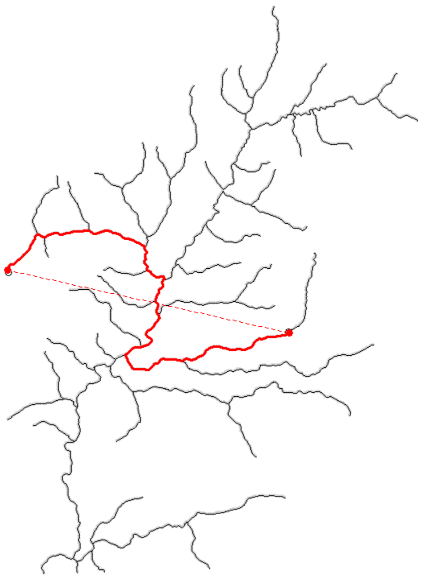

Figure 2: showpath

Fig. 4. Network path between two arbitrary points, displayed using `showpath`. The dashed line corresponds to the Euclidean distance. Euclidean and network distances are reported.

The network graph used by `networkdistance` is typically that generated by `read.linearmask` and saved as the 'graph' attribute of a linearmask. Network distance is then the sum of distances between adjacent mask points along the shortest network path. The distance is approximate because –

1. A simple adjacency rule is used to construct the network from the mask points, supplemented by the terminal points of lines. Points are considered 'adjacent' if their Euclidean separation is less than the mask 'spacing' times the mask attribute 'spacingfactor' (default 1.5). The graph formed using this rule mirrors the linear network except for occasional shortcut 'skips' where lines meet obliquely.
2. The distance is based on discretization of the linear network as a set of segments represented by their centroids. Incomplete terminal segments are dropped if they occur at the end of a Lines object. The component Line objects within a Lines object (Pebesma and Bivand 2005) are effectively placed end-on-end, and centroids are spaced equally along the combined length.

The convenience of the discretized network generally outweighs the cost of any imprecision, especially when the mask spacing is small. The risk of skips increases with the spacing factor. Reducing the spacing factor increases the risk of introducing breaks in the network.

The user may replace the default graph with one that more precisely represents the linear network. For example, nodes may be added precisely at the intersections of streams, and linked by edges to the adjoining mask points, while deleting 'skip' edges. You are mostly on your own here, but see https://rpubs.com/edzer/spatialnetworks/ and possibly https://CRAN.R-project.org/package=shp2graph/.

The functions `showedges`, `replot` and `deleteedges` are provided for editing the graph (see Examples in `help(deleteedges)`). `addedges` may be used to replace edges deleted by mistake or to bridge unwanted gaps in a network. `cleanskips` drops all but the shortest edge joining any two different lines; it is called by default by `read.linearmask`.

Detector layouts

Detector layouts ('traps' objects in **secr**) are the same in linear and 2-dimensional habitat models, except for the fact that detectors are located along the linear habitat features. There is no special 'lineartraps' class.

Input formats for detector layouts are described in secr-datainput.pdf.

The function `make.line` generates a traps object with detectors spaced equally along each line in a linear mask, or in clustered patterns, possibly with a random offset as shown here:

```r
trps <- make.line(linmask, detector = "proximity", n = 40, startbuffer = 0, by = 300,
                  endbuffer = 80, cluster = c(0,40,80), type = 'randomstart')
```

```r
plot(linmask)
plot(trps, add = TRUE, detpar = list(pch = 16, cex = 1.5, col='red'))
```

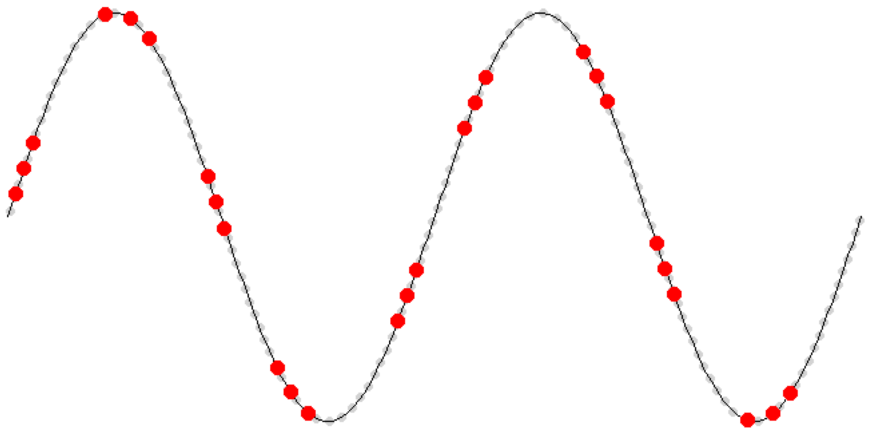

Figure 3: detectors

Fig. 5. Detectors placed with `make.line`

`make.line` places detectors on each component line of the mask, independently of other lines. The 'random-start' option is not guaranteed to provide a spatially representative sample. You may also locate detectors in the field by GPS and import the data with `read.traps`. For ad hoc tests, detectors may be placed just by clicking on a map:

```r
plot(silverstreammask)
loc <- locator(30)
xy <- snapPointsToLinearMask(data.frame(loc), silverstreammask)
tr <- read.traps(data = xy, detector = 'multi')
plot(tr, add = TRUE)
```

For distance calculations in **secrlinear**, the location of each detector is 'snapped' to the nearest mask point.

This behaviour is slightly undesirable because it limits network distances to discrete values (usually multiples of the mask spacing), but it should have little effect as long as mask spacing is much less than detector spacing.

The 'graph' attribute this will be used by `networkdistance` to compute distances. Edges should have a 'weight' attribute equal to the distance between centroids (equal to the pixel size for most pixel pairs, but no all).

Only point detector types are allowed. This excludes area search (polygon, polygonX) and transect search (transect, transectX). Data are commonly collected by searching a linear habitat (e.g., using dogs to search for scat along riverbanks). Model such data by discretizing the searched transect(s) as a series of point detectors. There are several ways to do this. Probably the most straightforward is to import both the transect track(s) and the point detections to **secr** as a capthist object with detector type 'transect'; the transect(s) may then be cut into equal segments with the `snip` function, and the snipped transects converted to point detectors with the `reduce` method for capthist data. The x-y coordinates for the centre of each segment may also be generated manually or with the **secrlinear** function `make.line`, but this requires more effort to match detections to searched segments. Data from searches most likely are of detector type 'count' to allow a Poisson number of detections per animal per occasion per segment.

This code shows the steps to take; see secr-datainput.pdf for data formats.

```r
transects <- read.traps('transectxy.txt', detector = 'transect')
capt <- read.table('capt.txt')
tempCH <- make.capthist(capt, transects, fmt = 'XY')
tempCH <- snip(tempCH, by = 100)   # for 100-m segments
CH <- reduce(tempCH, outputdetector = "count")
```

Simulating detection data

Here we describe the core functions used to simulate detection data for linear habitats. See also Evaluating study designs.

`sim.linearpopn` generates a population of a known density distributed at random along a linear mask. This is merely a simple wrapper for the more elaborate **secr** function `sim.popn`.

Further, the **secr** function `sim.capthist` may be used to generate detection datasets ('capthist' objects) from a specified population and detector layout ('traps' object). (This capability is used in the help files to fake data for demonstrating other functions).

For 2-dimensional habitat, `sim.capthist` will generate a population automatically if none is provided. For linear habitat, the user must explicitly simulate a population and supply this as the value of the 'popn' argument of `sim.capthist`.

The linear mask supplied to `sim.linearpopn` is retained as an attribute of the simulated population and used in `sim.capthist` as an argument of the supplied 'userdist' function. This enables detection probabilities to be modelled in `sim.capthist` using network the distance between each detector and an animal's home-range centre.

Here's how it works, carrying on our Silverstream example:

```r
# simulate population of 2 animals / km
pop <- sim.linearpopn(mask = silverstreammask, D = 2)
# simulate detections using network distances
CH <- sim.capthist(traps = tr, popn = pop, noccasions = 4,
                   detectpar = list(g0 = 0.25, sigma = 500),
                   userdist = networkdistance)
summary(CH)    # detector spacing uses Euclidean distances
```

```
## Object class       capthist
## Detector type      multi (4)
## Detector number    30
## Average spacing    456.6 m
## x-range            1397612 1402450 m
## y-range            4919286 4928129 m
##
## Counts by occasion
##                     1  2  3  4 Total
## n                  20 20 18 16    74
## u                  20  8  7  4    39
## f                  19 10  5  5    39
## M(t+1)             20 28 35 39    39
## losses              0  0  0  0     0
## detections         20 20 18 16    74
## detectors visited 14 15 15 13    57
## detectors used     30 30 30 30   120
##
## Individual covariates
##   sex
##   F:27
##   M:12
```

```r
# and plot the simulated detections...
par(mar = c(1,1,1,1))
plot(silverstreammask)
plot(CH, add = TRUE, tracks = TRUE, varycol = TRUE, rad = 100, cappar = list(cex = 2))
plot(tr, add = TRUE)
```

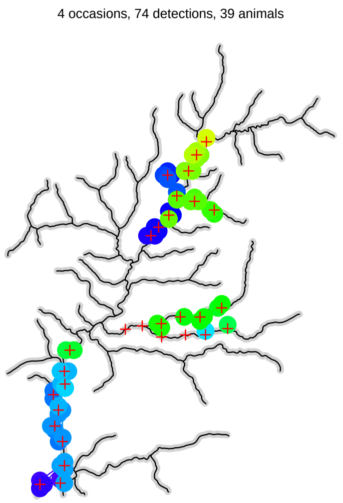

Fig. 6. Simulated detections of linear population. Colours vary (subtly) among individuals. Red crosses indicate trap locations. Movements ('tracks') are shown as-the-crow-flies rather than along the network.

Model fitting

Spatially explicit capture–recapture models for linear populations may be fitted using the **secr** function `secr.fit` with only slight alterations:

1. A linear mask is provided as the 'mask' argument.

2. Network distances are used instead of Euclidean distances.

Network distances may be specified either by passing a pre-computed matrix of distances between each trap (rows) and each mask point (columns), or by passing the function `networkdistance` that will compute the matrix on-the-fly. Either the function or the matrix is provided as the value of the 'details' component 'userdist' e.g., `details = list(userdist = networkdistance)`. Continuing the Silverstream example,

```
userd <- networkdistance(tr, silverstreammask)
userd[!is.finite(userd)] <- 1e8  # testing
sfit <- secr.fit(CH, mask = silverstreammask, details = list(userdist = userd))
predict(sfit)
```

Linear habitat models differ from 2-dimensional models in the interpretation of the parameters. Population density (D) is expressed as the expected or realised number of animals per kilometer of habitat, rather than per hectare. The detection function represents the probability of detection (g) or the expected number of detections (λ) at a given *network distance* from an animal's home-range centre. In other words, the fitted parameter σ scales with network distance (the units for σ are metres, as with a 2-dimensional model).

Advanced topics

Population size and effective sampling area

The usual **secr** functions for population size in a defined region, and for derived (conditional-likelihood) estimates, may be applied to 1-D models as for 2-D models, but the interpretation of the output differs. We can see this with the water vole models fitted earlier:

```
region.N(fit2DEuc)
```

```
##     estimate SE.estimate   lcl   ucl  n
## E.N    58.29       13.87 36.79 92.34 26
## R.N    58.29       11.58 42.33 89.86 26
```

```
region.N(fit1DNet)
```

```
##     estimate SE.estimate   lcl   ucl  n
## E.N    34.59       6.874 23.52 50.87 26
## R.N    34.59       3.558 29.94 44.74 26
```

The region of interest and the population model have been defined quite differently for 1-D and 2-D, and it is no surprise that the estimated populations are also different. For 2-D, the region is the original mask (area 45.5 ha) and the estimated density is extrapolated across its full extent although we believe animals are confined to the riverbank: this makes little sense. For 1-D, the region is the length of 'glymemask' along the river (1.3 km).

```
par(mfrow = c(1,2), mar = c(1,1,1,1))
plot(fit2DEuc$mask)
plot(traps(arvicola), add = TRUE)
mtext(side = 3,line = -1.5, "fit2DEuc$mask", cex = 1)
plot(fit1DNet$mask)
plot(traps(arvicola), add = TRUE)
mtext(side = 3,line = -1.5,"fit1DNet$mask", cex = 1)
```

Fig. 7. Comparison of 2-D and 1-D masks for water voles on R. Glyme. Traps in red.

Comparison of results from the `derived` function is also instructive:

```
derived(fit2DEuc)
```

```
##     estimate SE.estimate    lcl    ucl     CVn    CVa     CVD
```

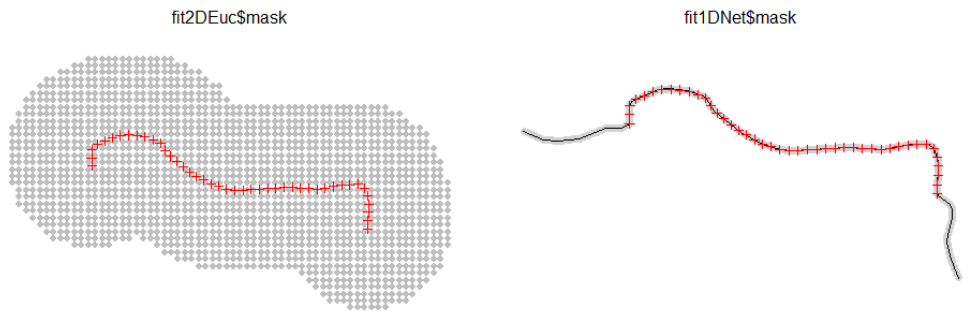

Figure 4: mask

```
## esa    20.32          NA     NA    NA     NA    NA     NA
## D       1.28      0.3004 0.8129 2.015 0.1961 0.129 0.2347
```

```
derived(fit1DNet)
```

```
##     estimate SE.estimate  lcl   ucl    CVn     CVa    CVD
## esa   0.9802          NA   NA    NA     NA      NA     NA
## D    26.5252       5.221 18.1 38.87 0.1961 0.01679 0.1968
```

In the 1-D model 'esa' (or a) refers to 'effective linear extent of sampling' rather than 'effective sampling area', and the units are km. The relative precision of $\hat{a}$ (CVa) is very much better for the 1-D model than the 2-D model, and this improves the precision of the density estimate. The other component of precision (CVn) is identical for the two models when `distribution = "poisson"` (the default).

Linear habitat covariates

Mask covariates are used to model spatial variation in population density. If the input to `read.linearmask` is a SpatialLinesDataFrame then its line-level attributes become covariates of the final mask. Covariates may also be added either directly, by assigning new columns in the 'covariates' dataframe, or indirectly, using the **secr** function `addCovariates`. `addCovariates` extracts data for each mask point from another spatial data source; this will usually be a SpatialPolygonsDataFrame or 2-dimensional mask object.

Suppose we wish to model density as a function of distance from the main channel (spine). In this code we interactively identify the multiple lines comprising the spine and compute a new column for the covariates dataframe.

```
# interactively obtain LineID for central 'spine' by clicking on
# each component line in plot
tmp <- getLineID(silverstreammask)
# extract coordinates of 'spine'
spine <- subset(silverstreammask, LineID = tmp$LineID)
# obtain network distances to spine and save for later use
netd <- networkdistance(spine, silverstreammask)  # matrix dim = c(nrow(spine), nrow(mask))
dfs <- apply(netd, 2, min) / 1000  # km
covariates(silverstreammask)$dist.from.spine <- dfs
```

Now let's plot the covariate –

```
par(mar=c(1,1,1,4))
plot(silverstreammask, covariate = 'dist.from.spine', col = topo.colors(13),
     cex = 1.5, legend = FALSE)
strip.legend('right', legend = seq(0, 6.5, 0.5), col = topo.colors(13),
             title = 'dist.from.spine km', height = 0.35)
plot(spine, add = TRUE, linecol = NA, cex = 0.3)
```

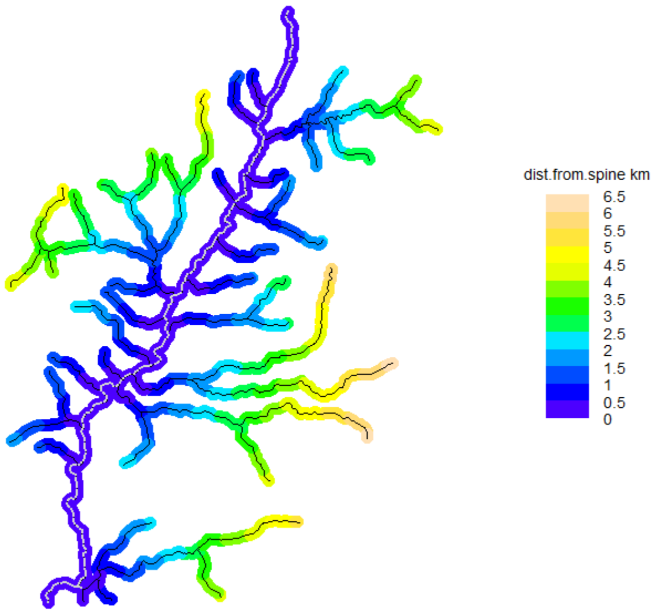

Figure 5: disttospine

Fig. 8. Computed covariate of Silverstream mask points – distance in km from central channel (spine)

The covariate may appear later as a predictor of density in `secr.fit` formulae (e.g., D ~ dist.from.spine). Remember that if you wish to predict detection parameters (g0, sigma etc.) the relevant covariate(s) generally should be associated with detectors (the 'traps' component of a 'capthist' object).

Discontinuities and traversable non-habitat

'Habitat' is the set of places where an animal may live, which we interpret for SECR as the set of possible home-range centres (loosely defined). Under the usual 2-dimensional SECR model, detection is a function of distance between a detector and the home-range centre irrespective of the continuity of habitat. This entails the (usually unstated) assumption that non-habitat is traversable, and detection is not prevented by intervening non-habitat. The reverse assumption (non-habitat is not traversed) is the default in linear SECR models. This is because distance is calculated from the network graph, and mask points that are not connected, either directly or indirectly, are considered to be an infinite distance apart.

Detections of an individual at unconnected detectors are strictly impossible under the default model, and such data will cause `secr.fit` to fail. This may result simply from data entry errors, or from a local failure of the algorithm used by `read.linearmask` to construct the graph. The function `checkmoves` finds animals whose sequential re-detection distances ('moves') lie outside a specified acceptable range:

```
# initially OK (no movement > 1000 m)--
checkmoves(arvicola, mask = glymemask, accept = c(0,1000))
# deliberately break graph of linear mask
attr(glymemask, 'graph')[200:203,201:204] <- NULL
# no longer OK --
```

```
out <- checkmoves(arvicola, mask = glymemask, accept = c(0,1000))
# display captures of animals 32 and 35 whose records span break
out$df
```

```
# problem shows up where voles recaptured either side of break:
showedges(glymemask, col = 'red', lwd = 6)
plot(out$CH, add = TRUE, tracks = TRUE, rad=8,cappar=list(cex=1.5))
pos <- traps(arvicola)['560.B',]
text(pos$x+5, pos$y+80, 'break', srt=90, cex=1.1)
```

Where individual detection histories are found to span a linear habitat gap, or for biological reasons we expect them to do so, the default network graph is an inadequate model for movement. It may be sufficient to 'bridge' breaks in the graph by adding edges manually:

```
plot(glymemask)
replot(glymemask)  # click on corners to zoom in
showedges(glymemask, col = 'red', lwd = 2, add=T)
glymemask <- addedges(glymemask)
```

The edge weight (edge length) assigned by `addedges` defaults to the Euclidean geographical distance between pixel centroids, which is an adequate approximation for small distances. The appropriate weight for edges added to bridge zones of non-habitat is a larger question that will depend on the biology of the species, and may involve estimated non-Euclidean distance parameters as in Royle et al. (2013) and secr-noneuclidean.pdf.

Linear home-range size

The implementation of the SECR model in **secrlinear** uses σ as the parameter to represent the spatial scale of detection. For many purposes σ may also be interpreted as the spatial scale of animal movements, or an index of home-range size. However, there are subtleties in the relation between σ and home-range size for both 2-dimensional and linear habitat. Two animals with the same value of σ will have the same home-range size (perhaps defined as a 95% activity area) only if there is an equal extent of habitat within their respective neighbourhoods. This need not be the case for individuals in a dendritic network (Fig. 9).

```
par(mfrow = c(1,1), mar = c(1,1,1,5))
plot(silverstreammask)
centres <- data.frame(locator(4))
OK <- networkdistance(centres, silverstreammask) < 1000
for (i in 1:nrow(OK)) {
    m1 <- subset(silverstreammask, OK[i,])
    plot(m1, add = TRUE, col = 'red', cex = 1.7)
    ml <- masklength(m1)
    points(centres, pch = 16, col = 'yellow', cex = 1.4)
    text (1406000, mean(m1$y), paste(ml, 'km'), cex = 1.2)
}
```

Fig. 9. Varying lengths of stream within 1 km of four arbitrarily located home range centres (yellow dots) (distances measured along network). Stream length associated with each centre depends on local habitat geometry.

It is possible, in principle, to fit a model in which σ varies across the network inversely with the channel density, so as to maintain a constant length of channel in each animal's home range.

Evaluating study designs

The R package **secrdesign** is a convenient wrapper for the simulation capability of **secr**, and this extends to linear habitats. Use **secrdesign** to assess potential study designs with respect to measures such as the likely

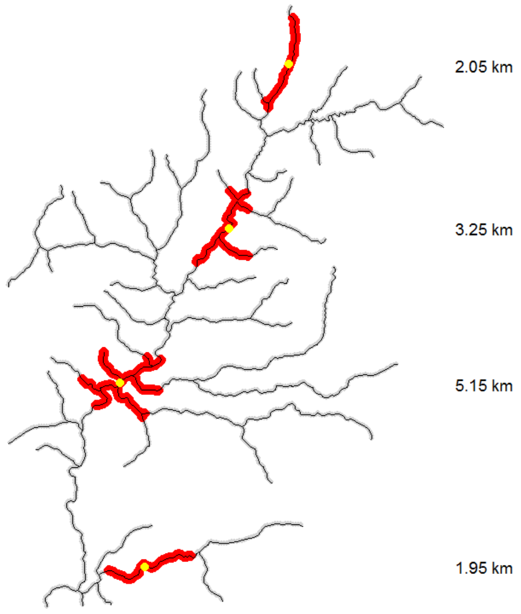

Figure 6: networklength

number of detections and repeat detections, and the expected bias and precision of density estimates from a fitted model. Here is a simple example using two linear detector layouts with a population at two densities (50/km, 200/km) in a linear habitat.

```
library(secrdesign)
```

```
## This is secrdesign 2.5.11. For overview type ?secrdesign
```

```
# create a habitat geometry
x <- seq(0, 4*pi, length = 200)
xy <- data.frame(x = x*100, y = sin(x)*300)
linmask <- read.linearmask(data = xy, spacing = 5)

# define two possible detector layouts
trp1 <- make.line(linmask, detector = "proximity", n = 80, start = 200, by = 30)
trp2 <- make.line(linmask, detector = "proximity", n = 40, start = 200, by = 60)
trplist <- list(spacing30 = trp1, spacing60 = trp2)

# create a scenarios dataframe
scen1 <- make.scenarios(D = c(50,200), trapsindex = 1:2, sigma = 25, g0 = 0.2)

# we specify the mask, rather than construct it 'on the fly',
# we will use a non-Euclidean distance function for both
# simulating detections and fitting the model...
det.arg <- list(userdist = networkdistance)
fit.arg <- list(details = list(userdist = networkdistance))

# run the scenarios and summarise results
sims1 <- run.scenarios(nrepl = 50, trapset = trplist, maskset = linmask,
    det.args = list(det.arg), fit.args = list(fit.arg),
    scenarios = scen1, seed = 345, fit = FALSE)
```

```
## Completed scenario 1

## Completed scenario 2

## Completed scenario 3

## Completed scenario 4

## Completed in 0.264 minutes
```

```
summary(sims1)
```

```
## run.scenarios(nrepl = 50, scenarios = scen1, trapset = trplist,
##     maskset = linmask, det.args = list(det.arg), fit = FALSE,
##     fit.args = list(fit.arg), seed = 345)
##
## Replicates    50
## Started       11:32:37 04 May 2021
## Run time      0.264  minutes
## Output class  selectedstatistics
##
## $constant
##              value
## noccasions       3
## nrepeats         1
## g0             0.2
## sigma           25
## detectfn         0
## recapfactor      1
## popindex         1
## detindex         1
## fitindex         1
## maskindex        1
##
## $varying
##  scenario trapsindex   D
##         1          1  50
##         2          2  50
##         3          1 200
##         4          2 200
##
## $detectors
##  trapsindex trapsname
##           1 spacing30
##           2 spacing60
##
## $det.args
##  detindex   userdist
##         1 userdistfn
##
## $fit.args
##  fitindex             details
##         1 userdist=userdistfn
##
```

```
## OUTPUT
##
## $1
##  1
##         n    mean      se
## n    50 93.0400 1.26973
## r    50 64.2400 1.64616
## nmov 50 48.5400 1.40849
## dpa  50  1.4573 0.01058
## rse  50  0.1262 0.00156
##
## $2
##  2
##         n    mean      se
## n    50 57.7400 1.03285
## r    50 16.1600 0.67694
## nmov 50  6.4200 0.43621
## dpa  50  1.0999 0.00604
## rse  50  0.2576 0.00585
##
## $3
##  3
##         n      mean      se
## n    50 358.18000 2.40487
## r    50 246.46000 3.09657
## nmov 50 189.48000 2.69909
## dpa  50   1.46062 0.00549
## rse  50   0.06389 0.00041
##
## $4
##  4
##         n     mean      se
## n    50 234.3400 2.01780
## r    50  64.5800 1.11578
## nmov 50  26.6200 0.65209
## dpa  50   1.1030 0.00244
## rse  50   0.1251 0.00108
```

Given some more time, we could set 'fit = TRUE' and assess the bias and precision of density estimates from the four levels of sampling intensity crossed with density.

```
sims2 <- run.scenarios(nrepl = 5, trapset = trplist, maskset = linmask,
     det.args = list(det.arg), scenarios = scen1, seed = 345, fit = TRUE)
summary(sims2)
```

Limitations, tips and troubleshooting

1. `read.linearmask` treats each Lines component of a SpatialLinesdataFrame as a separate entity, and discretizes it independently of other lines. The input (even a SpatialLinesDataFrame) has no inherent topology (junctions are not coded). The discretized form has an integer number of equal-length line segments for each Lines component in the input. This has a minor distorting effect on network distances. Each 'Lines' component may be composed of several 'line' objects, and these need not connect. However, from version 1.0.2 onwards, terminal fragments are 'run-on' to the start of the next 'line' within the same 'Lines' object, reducing the potential truncation error.

2. Many functions in **secr** for manipulating 2-dimensional detector arrays or model fits are unsuitable for models fitted with a linear mask, and in some cases a linear analogue has yet to be programmed. Examples are `buffer.contour`, `fxi.contour`, `fxi.total`, `circular.p`, `trap.builder`, `cluster.centres`, `distancetotrap`, `nearesttrap`, `plot.Dsurface`, and `pdot.contour`. Many other functions have not been tested with linear inputs: please report problems.
3. `ip.secr` has yet to be adapted for user-provided (non-Euclidean) distances, and its default trap-revealed home range measure (`RPSV`) is undefined for non-Euclidean distances, including network distances.
4. Area-search methods (polygon detector types) are not appropriate.
5. Linear search (transect detector types) are appropriate, and would be useful, but have yet to be implemented. The problem is that the numerical integration algorithm used in C code to predict overlap between home ranges and the searched transect requires that distances be computed on-the-fly for arbitrary locations, rather than from a pre-computed matrix.
6. Two-dimensional masks are often constructed by placing a 'buffer' around the detector array. The appropriate buffer width is a multiple of the movement scale σ, perhaps 4σ. A similar procedure may be followed for linear masks. We note that when movements (and home ranges) are strictly linear, a smaller multiple of σ is needed to encompass a given fraction of the animal's active locations. While a radius of 2.45σ encompasses 95% of a circular bivariate normal home range, a span of $\pm1.96\sigma$ spans the same fraction of a 2-sided linear normal home range.

References

Bivand, R., Keitt, T and Rowlingson, B. (2016). rgdal: Bindings for the Geospatial Data Abstraction Library. R package version 1.1-9. https://CRAN.R-project.org/package=rgdal/

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

Csardi, G. and Nepusz, T. (2006) The igraph software package for complex network research, InterJournal, Complex Systems 1695. https://igraph.org/.

Efford, M. G. (1985) *The structure and dynamics of water vole populations.* D.Phil thesis, University of Oxford.

Efford, M. G. and Mowat, G. (2014) Compensatory heterogeneity in spatially explicit capture–recapture data. *Ecology* **95**, 1341–1348.

ESRI (1998) *ESRI shapefile technical description.* https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

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

Royle, J. A., Chandler, R. B., Gazenski, K. D. and Graves, T. A. (2013) Spatial capture–recapture models for jointly estimating population density and landscape connectivity. *Ecology* **94** 287–294.

Sutherland, C., Fuller, A. K. and Royle, J. A. (2015) Modelling non-Euclidean movement and landscape connectivity in highly structured ecological networks. *Methods in Ecology and Evolution* **6**, 169–177.

Appendix. Extended water vole example.

Here we continue the water vole analysis with one or two twists... The 'hcov' formulation estimates the sex ratio and allows for sex-based models of detection (see ?hcov).

We assume the data have been prepared as in the introductory example.

```
# It is efficient to pre-compute a matrix of distances between traps (rows)
# and mask points (columns)
distmat <- networkdistance (traps(arvicola), glymemask, glymemask)

# Morning and evening trap checks as a time covariate
tcov <- data.frame(ampm = rep(c("am","pm"),3))

glymefit1 <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                      details = list(userdist = distmat),
                      model = g0~1, hcov = "sex")
glymefit2 <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                      details = list(userdist = distmat),
                      model = g0~ampm, timecov = tcov, hcov = "sex")
glymefit3 <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                      details = list(userdist = distmat),
                      model = g0~ampm + h2, timecov = tcov, hcov = "sex")
glymefit4 <- secr.fit(arvicola, mask = glymemask, trace = FALSE,
                      details = list(userdist = distmat),
                      model = list(sigma~h2, g0~ampm + h2),
                      timecov = tcov, hcov = "sex")

fitlist <- secrlist(glymefit1, glymefit2, glymefit3, glymefit4)
# dropping the detectfn (halfnormal) column to save space...
AIC(fitlist)[,-2]
#                                       model npar logLik   AIC  AICc dAICc AICcwt
# glymefit4 D~1 g0~ampm + h2 sigma~h2 pmix~h2    7 -322.5 659.1 665.3  0.00      1
# glymefit3  D~1 g0~ampm + h2 sigma~1 pmix~h2    6 -347.3 706.7 711.1 45.80      0
# glymefit2       D~1 g0~ampm sigma~1 pmix~h2    5 -353.5 717.0 720.0 54.66      0
# glymefit1          D~1 g0~1 sigma~1 pmix~h2    4 -356.8 721.6 723.5 58.20      0

# summaries of estimated density and sex ratio under different models
options(digits=3)

# model does not affect density estimate
collate(fitlist, perm = c(2,3,1,4))[,,1,"D"]
#           estimate SE.estimate  lcl  ucl
# glymefit1     26.5        5.27 18.0 39.0
# glymefit2     26.4        5.26 18.0 38.9
# glymefit3     26.3        5.25 17.9 38.8
# glymefit4     27.2        5.45 18.5 40.2

# model does affect the estimate of sex ratio (here proportion female)
collate(fitlist, perm=c(2,3,1,4))[,,1,"pmix"]
#           estimate SE.estimate   lcl   ucl
# glymefit1    0.615      0.0954 0.421 0.779
# glymefit2    0.615      0.0954 0.421 0.779
# glymefit3    0.634      0.0938 0.439 0.793
# glymefit4    0.669      0.0897 0.477 0.817

# predictions from best model
newdata <- expand.grid(ampm = c("am", "pm"), h2 = c("F", "M"))
predict(glymefit4, newdata = newdata)
```

```
# $`ampm = am, h2 = F`
#         link estimate SE.estimate    lcl    ucl
# D        log   27.239      5.4478 18.477 40.158
# g0     logit    0.218      0.0463  0.141  0.322
# sigma    log   13.624      1.8764 10.414 17.823
# pmix   logit    0.669      0.0897  0.477  0.817
#
# $`ampm = pm, h2 = F`
#         link estimate SE.estimate     lcl    ucl
# D        log   27.239      5.4478 18.4768 40.158
# g0     logit    0.116      0.0293  0.0694  0.186
# sigma    log   13.624      1.8764 10.4136 17.823
# pmix   logit    0.669      0.0897  0.4774  0.817
#
# $`ampm = am, h2 = M`
#         link estimate SE.estimate     lcl    ucl
# D        log   27.239      5.4478 18.4768 40.158
# g0     logit    0.153      0.0392  0.0908  0.246
# sigma    log   70.958     10.0551 53.8247 93.545
# pmix   logit    0.331      0.0897  0.1829  0.523
#
# $`ampm = pm, h2 = M`
#         link estimate SE.estimate     lcl    ucl
# D        log  27.2394      5.4478 18.4768 40.158
# g0     logit   0.0782      0.0201  0.0468  0.128
# sigma    log  70.9581     10.0551 53.8247 93.545
# pmix   logit   0.3311      0.0897  0.1829  0.523
```

Some general observations:

The likelihood used is strictly for a multi-catch trap type (detector = "multi" rather than detector = "single"), but given the low trap saturation we do not expect this to cause any problem.

Water voles are active both day and night. The greater g0 for morning trap checks is explained by the longer interval between the afternoon and morning checks.

Precision is poor (RSE about 20%). If we were concerned only with the realised density on this river we would be justified in setting distribution = "binomial"'; this improves the RSE ('CVD') to about 10%:

```
derived(glymefit4, distribution = 'binomial')
#     estimate SE.estimate   lcl   ucl    CVn     CVa    CVD
# esa   0.9545          NA    NA    NA     NA      NA     NA
# D    27.2396       2.867 22.17 33.46 0.1038 0.01747 0.1053
```

Adult males had much larger home ranges than adult females at this time of year (many females were suckling young in burrows; males search out and fight over oestrous females).

Reciprocal variation in g0 and sigma between the sexes to some extent mitigated the effect of sex differences on density estimates (cf Efford and Mowat Ecology 2014).

The terminal "buffer" on the linear mask (200–240 m beyond the terminal traps) was adequate, even for males.