secrBVN - simulation of spatially explicit capture-recapture with bivariate normal home ranges

Murray Efford

2018-07-20

Contents

This small package evaluates SECR when home ranges are BVN or uniform (flat-topped) ellipses.

The key user-visible functions are `simpopn.bvn`, `plotpopn.bvn`, `simcapt.bvn` and `runEllipseSim`.

Generating and plotting elliptical home ranges

`simpopn.bvn` is a wrapper for the **secr** function `sim.popn` that adds attributes specifying a bivariate normal home range shape, size and orientation for each individual. At first shape and size are the same for all individuals, but the resulting popn object may be modified so they vary individually.

First, load the package.

```
library(secrBVN)
```

```
## Loading required package: secr
```

```
## This is secr 3.1.7. For overview type ?secr
```

```
vignettefolder <- "c:/density secr 3.1/secrBVN/vignettes/"
```

```
tempgrid <- make.grid(nx = 10, ny = 10)
par(mfrow=c(2,4), mar = c(2,2,2.6,2), xpd = TRUE)
for (i in 1:4) {
    s2xy <- 25^2 * c(1/i, i)
    random.pop <- simpopn.bvn(s2xy = s2xy, core = tempgrid, buffer = 100, D = 1)
    plotpopn.bvn(random.pop, col = 'lightblue')
```

```
    mtext(side=3, line=1.5, i)
}
for (i in 1:4) {
    s2xy <- 25^2 * c(1/i, i)
    aligned.pop <- simpopn.bvn(s2xy = s2xy, core = tempgrid, buffer = 100, D = 1, theta = -1)
    plotpopn.bvn(aligned.pop, col = 'lightblue')
}
```

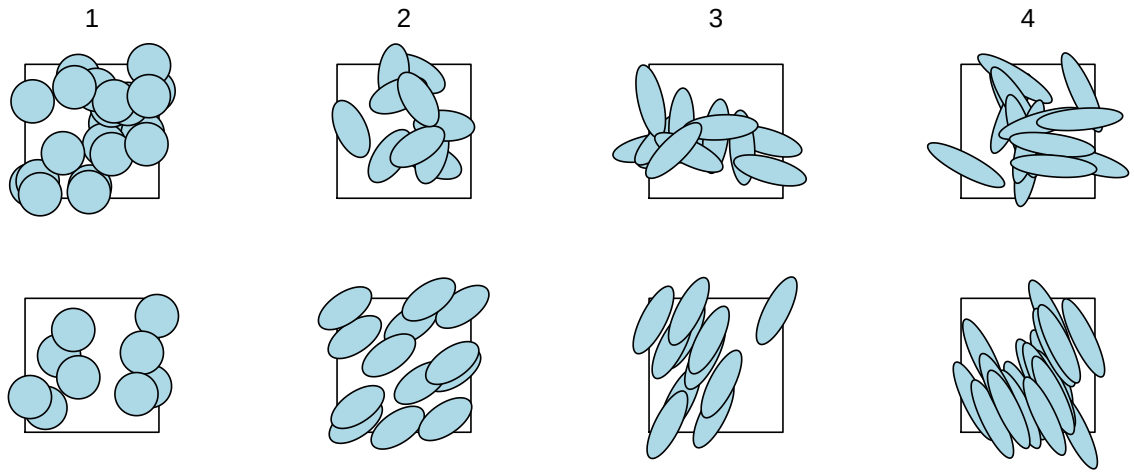


Fig. 1. Elliptical home ranges with varying ratio of major to minor axes as shown. Upper row oriented randomly and independently, lower row with shared random orientation ('randomly aligned').

Simulating

Generating detection histories

Normally in **secr** we use `sim.capthist` to generate capture histories, but that is limited to circular detection functions. The function `simcapt.bvn` is a partial replacement for `sim.capthist` that models detection with a bivariate normal. Specifically, the probability of detection is a constant times the bivariate normal probability density at the detector. The constant is $g_0 2\pi\sigma_X\sigma_Y$. The user provides a 'popn' object that includes the BVN parameter values $(\sigma_x^2, \sigma_y^2, \theta)$ for each animal (row), as generated above by `simpopn.bvn`. The constant scales the BVN density so that the maximum detection probability is g_0.

If `detectfn = 4` is selected then a uniform (flat-topped) elliptical home range is simulated.

Fitting a circular detection model to BVN data

The function `runEllipseSim` is a wrapper for the preceding steps (`simpopn.bvn`, `simcapt.bvn`) that also fits a standard (circular) SECR model with `secr.fit`. The default population has fixed number of individuals within the rectangular buffered area around the detectors (`Ndist = 'fixed'`).

Here we use a 6×6 grid of binary proximity detectors with 50 metre spacing. The code in **secrBVN** does not allow for competition among detectors (secr detector type 'multi') or other other secr detector types. A density of 4/ha gives exactly 169 animals in the buffered area. The intercept parameter g0 is varied to reduce the effect of `detectfn` on total number of captures and precision. Conditional likelihood is used for speed; the default `extractfn = derived` is compatible with both `CL = TRUE` and `CL = FALSE`. The 200-m buffer allows for the longest ranges ($\sigma_y = 50$ m).

```
nrepl <- 500
tr <- make.grid(6,6, spacing = 50, detector = 'proximity')
simrandom <- vector('list')
simrandomBVN <- vector('list')
for (i in 1:4) {
    sigmaX <- 25/i^0.5; sigmaY <- 25*i^0.5
    details <- list(distribution = 'binomial')
    ## uniform
    simrandom[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.2, D = 4, detectfn = 4, CL = TRUE,
                                details = details)
    ## bvn
    simrandomBVN[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
                                details = details)
}
save(simrandom, file = paste(vignettefolder, 'simrandom.RData', sep=''))
save(simrandomBVN, file = paste(vignettefolder, 'simrandomBVN.RData', sep=''))
```

Construct a function to summarize the results:

```
sumplot <- function (sims1, sims2, trueD = 4, xval = 1:4, ylim = c(-0.2,0.2),
                     legtext = c('setone' ,'settwo'), plt = TRUE) {
    sumD <- function(x) {
        Dval <- sapply(lapply(x, '[[', 'fit'), '[[', 'D', 'estimate')
        DSE  <- sapply(lapply(x, '[[', 'fit'), '[[', 'D', 'SE.estimate')
        Dlcl <- sapply(lapply(x, '[[', 'fit'), '[[', 'D', 'lcl')
        Ducl <- sapply(lapply(x, '[[', 'fit'), '[[', 'D', 'ucl')
        n <- sum(!is.na(Dval))
        RB <- (Dval-trueD)/trueD
        RSE <- DSE/Dval
        COV <- (trueD>=Dlcl) & (trueD<=Ducl)
        npop <- sapply(x, '[[', 'npop')
        nCH <- sapply(x, '[[', 'nCH')
        c(av.npop = mean(npop), av.nCH = mean(nCH), av.Dhat = mean(Dval),
          md.Dhat = median(Dval), sd.Dhat = sd(Dval),
          RB = mean(RB), seRB = sd(Dval)/trueD/n^0.5,
          RSE = mean(RSE), seRSE = sd(RSE)/n^0.5,
          COV = mean(COV))
    }
    out1 <- sapply(sims1, sumD)
    out2 <- sapply(sims2, sumD)
    if (plt) {
        plot(xval, out1['RB',], ylim = ylim, xlab = 'Aspect ratio',
             ylab = 'Relative bias', pch=16, axes = FALSE)
        axis(1, at=1:4)
        axis(2)
        segments(xval, out1['RB',]-2*out1['seRB',],
                 xval, out1['RB',]+2*out1['seRB',])
        segments(xval+0.05, out2['RB',]-2*out2['seRB',],
                 xval+0.05, out2['RB',]+2*out2['seRB',])
        points(xval+0.05, out2['RB',], pch = 21, bg = 'white')
        abline(h=0, lty=2)
        legend (par()$usr[2]*0.6, par()$usr[4]*0.95, legend = legtext,
```

```
                pch = c(16,21), cex = 0.8)
    }
    output <- list(out1,out2)
    names(output) <- legtext
    lapply(output,round,4)
}
```

. . . and run it:

```
load(file = paste(vignettefolder, 'simrandom.RData', sep=''))
load(file = paste(vignettefolder, 'simrandomBVN.RData', sep=''))
par(mfrow = c(1,1), mar = c(4,4,4,4), xpd = FALSE)
output <- sumplot(simrandom, simrandomBVN, legtext = c('Uniform','BVN'))
```

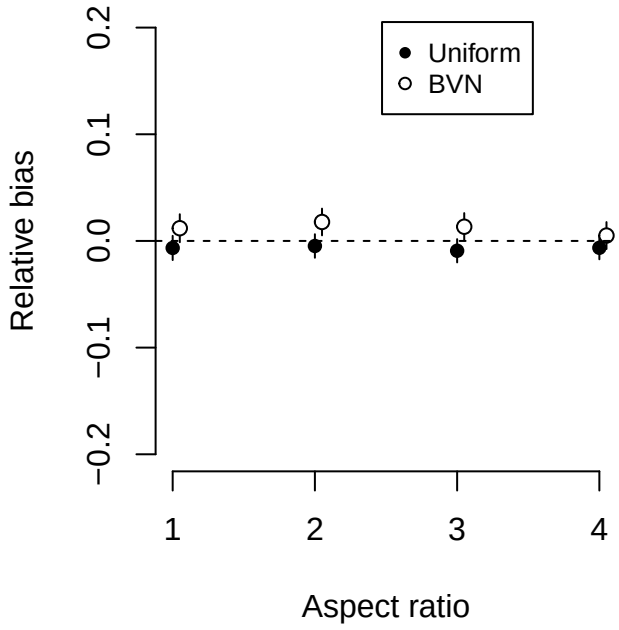


Fig. 2. Relative bias of density estimated by fitting circular SECR detection model to elliptical data, with 95% confidence limit for simulated values. 'Uniform' home ranges were truncated at the 95% activity contour of an equivalent bivariate normal, and detection probability was uniform inside the boundary. 'BVN' home ranges extended indefinitely in all directions.

```
output
```

```
## $Uniform
##             [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   49.4020  50.6700  52.3320  54.1020
## av.Dhat   3.9737   3.9811   3.9632   3.9743
## md.Dhat   3.9547   3.9465   3.9694   3.9693
## sd.Dhat   0.5078   0.4938   0.4881   0.4821
## RB       -0.0066  -0.0047  -0.0092  -0.0064
## seRB      0.0057   0.0055   0.0055   0.0054
## RSE       0.1217   0.1201   0.1183   0.1169
```

```
## seRSE     0.0003   0.0003   0.0003   0.0003
## COV       0.9300   0.9380   0.9360   0.9400
##
## $BVN
##              [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   43.0040  43.8700  44.8640  45.6120
## av.Dhat   4.0477   4.0711   4.0533   4.0202
## md.Dhat   4.0172   4.0231   4.0139   3.9712
## sd.Dhat   0.5865   0.5605   0.5713   0.5644
## RB        0.0119   0.0178   0.0133   0.0051
## seRB      0.0066   0.0063   0.0064   0.0063
## RSE       0.1359   0.1350   0.1342   0.1339
## seRSE     0.0004   0.0004   0.0004   0.0004
## COV       0.9400   0.9360   0.9280   0.9360
```

There is no apparent effect of range elongation itself on the bias of the estimates. Fitting a halfnormal detection function (detectfn = 0) to data from 'hard-edged' (uniform) home ranges (detectfn = 4) appears to result in negative bias on the order of –1% to –2% (Efford 2004 noted a relative bias of –1.2%, SE 0.8% for a small sample of 100 simulations fitting a model by inverse prediction in the circular case).

Limitations

This package has the limited goal of determining how range elongation affects estimates of density in simple SECR models, and these specific limitations:

1. Only binary proximity detectors are supported.
2. The spatial distribution of activity centres is assumed to be homogeneous Poisson.
3. 'g0' is used here loosely - probably it should be lambda0 (λ_0).
4. Ellipses are specified using either 'sigmaX' and 'sigmaY' as separate arguments (`runEllipseSim`) or as a vector of the two values, squared ('s2xy'). This is confusing but it's better at this point not to change.

References

Efford, M. G. (2004) Density estimation in live-trapping studies. *Oikos* **106**, 598–610.

Huggins, R. M. (1989) On the statistical analysis of capture experiments. *Biometrika* **76**, 133–140.

Ivan, J. S., White, G. C. and Shenk, T. M. (2013) Using simulation to compare methods for estimating density from capture–recapture data. *Ecology* **94**, 817–826.

Appendix. Supplementary simulations

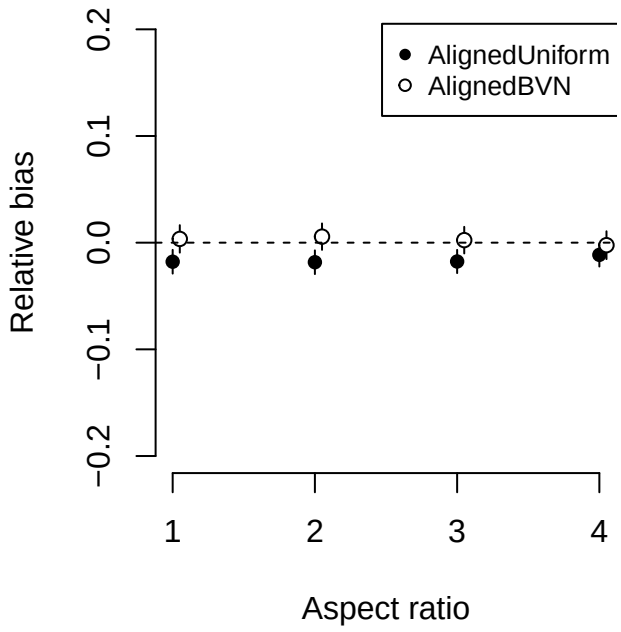
Randomly aligned ellipses

```
nrepl <- 500
tr <- make.grid(6,6, spacing = 50, detector = 'proximity')
simalign <- vector('list')
simalignBVN <- vector('list')
for (i in 1:4) {
```

```
    sigmaX <- 25/i^0.5; sigmaY <- 25*i^0.5
    details <- list(distribution = 'binomial')
    ## uniform
    simalign[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.2, D = 4, detectfn = 4, CL = TRUE,
                                details = details, theta = -1)
    ## bvn
    simalignBVN[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
                                details = details, theta = -1)
}
save(simalign, file = paste(vignettefolder, 'simalign.RData', sep=''))
save(simalignBVN, file = paste(vignettefolder, 'simalignBVN.RData', sep=''))
```

These results are not much different to those with random alignment, although the coverage is perhaps slightly worse.

```
load(file = paste(vignettefolder, 'simalign.RData', sep=''))
load(file = paste(vignettefolder, 'simalignBVN.RData', sep=''))
par(mfrow = c(1,1), mar = c(4,4,4,4), xpd = FALSE)
sumplot(simalign, simalignBVN, legtext = c('AlignedUniform','AlignedBVN'))
```



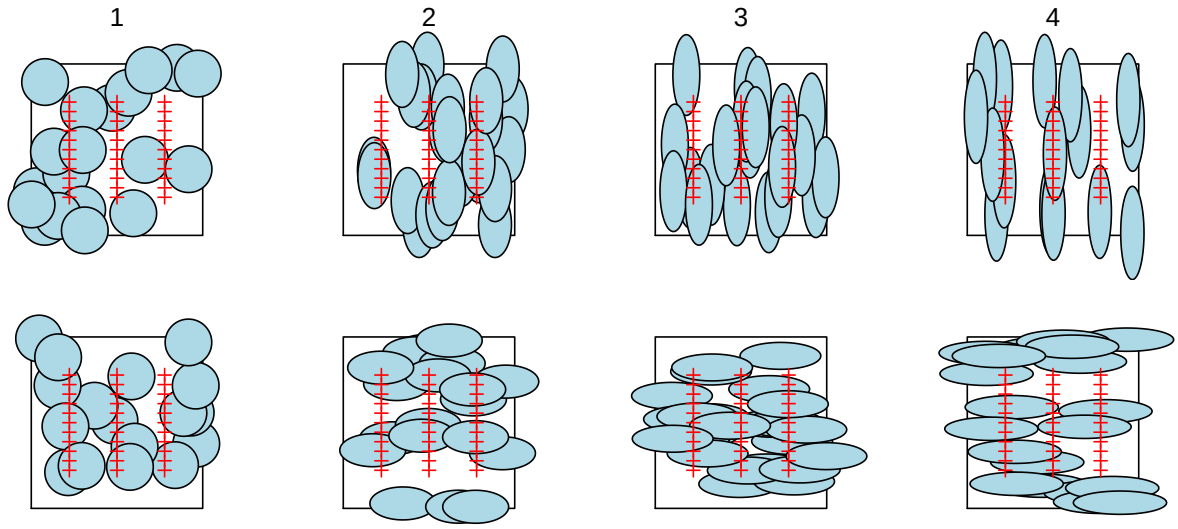

```
## $AlignedUniform
##             [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   48.7920  49.9720  51.8740  53.6740
## av.Dhat   3.9286   3.9267   3.9294   3.9543
## md.Dhat   3.9144   3.9264   3.9357   3.9667
## sd.Dhat   0.4985   0.4977   0.4873   0.4896
## RB       -0.0179  -0.0183  -0.0176  -0.0114
```

```
## seRB      0.0056   0.0056   0.0054   0.0055
## RSE       0.1224   0.1210   0.1189   0.1174
## seRSE     0.0003   0.0003   0.0003   0.0003
## COV       0.9280   0.9220   0.9280   0.9180
##
## $AlignedBVN
##             [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   42.6620  43.4480  44.4220  45.1500
## av.Dhat   4.0141   4.0226   4.0095   3.9906
## md.Dhat   4.0020   4.0299   3.9972   3.9711
## sd.Dhat   0.5761   0.5548   0.5582   0.5832
## RB        0.0035   0.0057   0.0024  -0.0023
## seRB      0.0064   0.0062   0.0062   0.0065
## RSE       0.1366   0.1357   0.1349   0.1347
## seRSE     0.0004   0.0004   0.0004   0.0004
## COV       0.9380   0.9280   0.9400   0.9200
```

Grid-aligned ellipses

Consider a scenario in which the trapping grid has a strong 'grain' (orientation) and elongated home ranges are aligned with the trapping grid.

```
tr <- make.grid(3,11, spacex = 125, spacey = 25, detector = 'proximity')
par(mfrow=c(2,4), mar = c(2,2,2.6,2), xpd = TRUE)
for (i in 1:4) {
    s2xy <- 25^2 * c(1/i, i)
    pop <- simpopn.bvn(s2xy = s2xy, core = tr, buffer = 100, D = 1, theta = 0)
    plotpopn.bvn(pop, col = 'lightblue')
    plot(tr, add=T)
    mtext(side=3, line=1.5, i)
}
for (i in 1:4) {
    s2xy <- 25^2 * c(1/i, i)
    pop <- simpopn.bvn(s2xy = s2xy, core = tr, buffer = 100, D = 1, theta = pi/2)
    plotpopn.bvn(pop, col = 'lightblue')
    plot(tr, add=T)
}
```

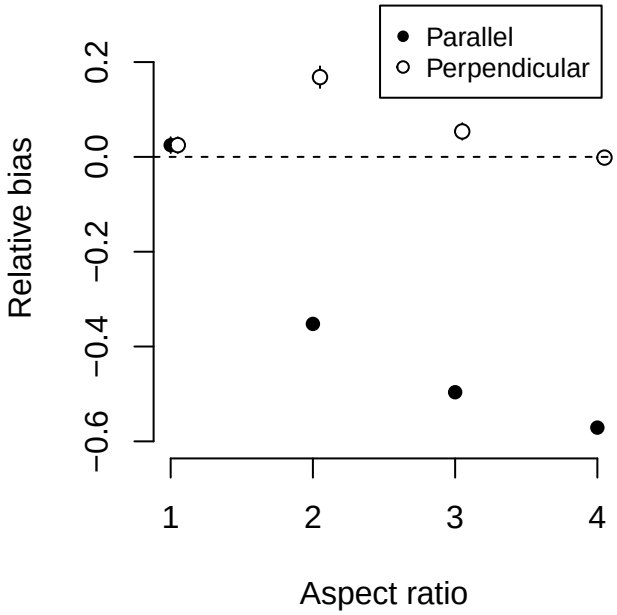



```
nrepl <- 500
tr <- make.grid(3,11, spacex = 125, spacey = 25, detector = 'proximity')
simgridalign1 <- vector('list')
simgridalign2 <- vector('list')
for (i in 1:4) {
    sigmaX <- 25/i^0.5; sigmaY <- 25*i^0.5
    details <- list(distribution = 'binomial')
    ## bvn parallel to traps
    simgridalign1[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
                                details = details, theta = 0)
    ## bvn perpendicular to traps
    simgridalign2[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 20,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
                                details = details, theta = pi/2)
}
save(simgridalign1, file = paste(vignettefolder, 'simgridalign1.RData', sep=''))
save(simgridalign2, file = paste(vignettefolder, 'simgridalign2.RData', sep=''))
```

```
load(file = paste(vignettefolder, 'simgridalign1.RData', sep=''))
load(file = paste(vignettefolder, 'simgridalign2.RData', sep=''))
```

These results are much different to those with random alignment and a square grid. Clearly this is an important design consideration. Recaptures are likely to be along a trap line when these are oriented similarly to home ranges and the spacing between trap lines is large.

```
par(mfrow=c(1,1), mar=c(4,4,4,4), xpd = FALSE)
sumplot(simgridalign1, simgridalign2, leg = c('Parallel','Perpendicular'), ylim = c(-0.6,0.3))
```




```
## $Parallel
##             [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   37.0800  31.1940  28.0440  26.2440
## av.Dhat   4.0994   2.5909   2.0144   1.7158
## md.Dhat   4.0369   2.5669   2.0087   1.7040
## sd.Dhat   0.7403   0.4623   0.3779   0.3498
## RB        0.0248  -0.3523  -0.4964  -0.5710
## seRB      0.0083   0.0052   0.0042   0.0039
## RSE       0.1748   0.1651   0.1651   0.1666
## seRSE     0.0007   0.0007   0.0007   0.0008
## COV       0.9400   0.2180   0.0080   0.0020
##
## $Perpendicular
##             [,1]     [,2]     [,3]     [,4]
## av.npop 169.0000 169.0000 169.0000 169.0000
## av.nCH   37.0800  42.2640  44.1160  45.1760
## av.Dhat   4.0994   4.6729   4.2147   3.9939
## md.Dhat   4.0369   4.5111   4.1332   3.9439
## sd.Dhat   0.7403   1.0193   0.7719   0.6360
## RB        0.0248   0.1682   0.0537  -0.0015
## seRB      0.0083   0.0114   0.0086   0.0071
## RSE       0.1748   0.1615   0.1477   0.1417
## seRSE     0.0007   0.0007   0.0006   0.0005
## COV       0.9400   0.8060   0.9020   0.9300
```

Anisotropic home ranges as a solution

In principle, we can deal with uniformly oriented and elongated ranges by replacing Euclidean distances with distances in a transformed space. Thanks to Ben Augustine for pointing the 'geoR' function `coords.aniso` that lets us do this. Whether this is a practical solution remains to be seen: if the parameters of the transformation are unknown and must be estimated (below) then intensive sampling may be needed to get good results.

This is the only time in this vignette that we attempt to *fit* elongated ranges rather than circular ranges.

```
anisodistfn <- function (xy1,xy2, mask) {
    if (missing(xy1)) return(character(0))
    xy1 <- as.matrix(xy1)
    xy2 <- as.matrix(xy2)
    miscparm <- attr(mask, 'miscparm')
    psiA <- miscparm[1]           ## anisotropy angle; identity link
    psiR <- 1 + exp(miscparm[2])  ## anisotropy ratio; log link
    aniso.xy1 <- geoR::coords.aniso(xy1, aniso.pars = c(psiA, psiR), reverse = FALSE)
    aniso.xy2 <- geoR::coords.aniso(xy2, aniso.pars = c(psiA, psiR), reverse = FALSE)
    distmat <- edist(aniso.xy1,aniso.xy2)
    distmat ## nrow(xy1) x nrow(xy2) matrix
}
nrepl <- 10
library(geoR)
tr <- make.grid(3,11, spacex = 125, spacey = 25, detector = 'proximity')
simaniso1 <- vector('list')
simaniso2 <- vector('list')
for (i in 1:4) {
    sigmaX <- 25/i^0.5; sigmaY <- 25*i^0.5
    ## parameters psiA and psiR are passed to anisodistfn
    details <- list(distribution = 'binomial', userdist = anisodistfn,
             miscparm = c(psiA = 0.5, psiR = 1.5))
#     ## bvn parallel to traps
#     simaniso1[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 1,
#                                 traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
#                                 details = details, theta = 0)
#     ## bvn perpendicular to traps
#     simaniso2[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 1,
#                                 traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
#                                 details = details, theta = pi/2)
    ## bvn parallel to traps
    simaniso1[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 1,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0,
                                extractfn = predict, details = details, theta = 0)
    ## bvn perpendicular to traps
    simaniso2[[i]] <- runEllipseSim (nrepl, sigmaX, sigmaY, buffer = 200, ncores = 1,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0,
                                extractfn = predict, details = details, theta = pi/2)
}
save(simaniso1, file = 'simaniso1.RData')
save(simaniso2, file = 'simaniso2.RData')
```

‘‘‘

Check circular using alternate method (sim.capthist)

```
nrepl <- 500
tr <- make.grid(6,6, spacing = 50, detector = 'proximity')
simcirc <- vector('list')
simcircBVN <- vector('list')
    sigmaX <- 25
    details <- list(distribution = 'binomial')
    ## uniform
    simcirc[[1]] <- runEllipseSim (nrepl, sigmaX, sigmaY=NULL, buffer = 200, ncores = 6,
                                traps = tr, g0 = 0.2, D = 4, detectfn = 4, CL = TRUE,
                                details = details)
    ## bvn
    simcircBVN[[1]] <- runEllipseSim (nrepl, sigmaX, sigmaY=NULL, buffer = 200, ncores = 6,
                                traps = tr, g0 = 0.4, D = 4, detectfn = 0, CL = TRUE,
                                details = details)
save(simcirc, file = 'simcirc.RData')
save(simcircBVN, file = 'simcircBVN.RData')
```

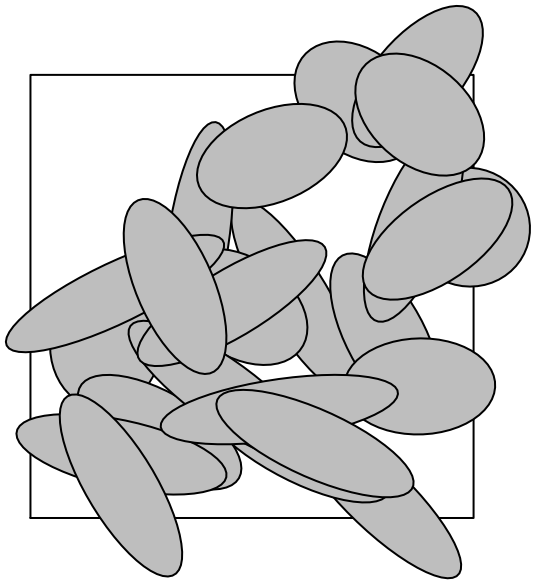
```
load(file = 'simcirc.RData')
load(file = 'simcircBVN.RData')
par(mfrow = c(1,1), mar = c(4,4,4,4), xpd = FALSE)
sumplot(simcirc, simcircBVN, legtext = c('Uniform','BVN'), plt = FALSE)
```

```
## $Uniform
##             [,1]
## av.npop 169.0000
## av.nCH   48.7560
## av.Dhat   3.9280
## md.Dhat   3.9124
## sd.Dhat   0.5032
## RB       -0.0180
## seRB      0.0056
## RSE       0.1225
## seRSE     0.0004
## COV       0.9240
##
## $BVN
##             [,1]
## av.npop 169.0000
## av.nCH   42.6780
## av.Dhat   4.0221
## md.Dhat   3.9949
## sd.Dhat   0.5989
## RB        0.0055
## seRB      0.0067
## RSE       0.1366
## seRSE     0.0005
## COV       0.9380
```

Heterogeneous elliptical home ranges

`simpopn.bvn` usually generates a population with equal-sized home ranges. The size and elongation of each range may be varied by providing a function as the argument `s2xy`:

```
tr <- make.grid(6,6, spacing = 50, detector = 'proximity')
rs2xy <- function(N, scale = 25) {
    aspectratio <- 1 + runif(N) * 3
    cbind(scale^2 / aspectratio, scale^2 * aspectratio)
}
pop <- simpopn.bvn(s2xy = rs2xy, core = tr, buffer = 100, D = 1)
par(mfrow = c(1,1), mar = c(4,4,4,4), xpd = TRUE)
plotpopn.bvn(pop, col='grey')
```



Heterogeneous populations may also simulated `runEllipseSim` by passing a function as the argument 'sigmaX'.

```
sims <- runEllipseSim (10, sigmaX = rs2xy, buffer = 200, ncores = 2,
                       traps = tr, g0 = 0.2, D = 4, detectfn = 4, CL = TRUE,
                       SECR = TRUE)
```