Efford, M. G. ipsecr: An R package for awkward spatial capture–recapture data. Methods in Ecology and Evolution

Appendix 1: Simulations with constant density and single-catch traps

Simulations were performed to validate the simulation and inverse prediction (IP) estimates from the R package 'ipsecr' for data from single-catch traps, and for comparison between methods. Seven density scenarios were simulated that resulted in a wide range of sample size and trap saturation (Table S1a); these scenarios were identical to those used by Efford (2004, Table 1).

Maximum likelihood estimates were obtained with the R package secr 4.5.8 (Efford 2022) and IP estimates were obtained with ipsecr 1.4.0, using default settings in function ipsecr.fit (Efford 2023a). Simulations were managed with the R package secrdesign 2.8.0 (Efford 2023b). R code for the simulations is provided below. 1000 replicate simulations were performed, reduced to 200 for IP estimates at higher densities. At the lowest density very few individuals were detected (often $n < 10$). At the lowest density, the IP method failed in 51 replicates out of 1000 and the ML method in 13 out of 1000, and at the next lowest density (1/ha) the IP estimator failed in 2 cases; estimation was otherwise successful.

The number of individuals caught (n) and trap saturation (TS) closely matched the earlier simulations for each level of population density (Table S1a). ML density estimates with the model for multi-catch traps were essentially unbiased, as found also by Efford et al. (2009), but estimates of g_0 for moderate to high trap saturation were highly biased (Table S1b). IP estimates of density showed a slight positive bias that was mostly negligible ($< 1\%$), but reached nearly 4% when captures were very sparse Table S1c). Coverage of 95% confidence intervals for D was close to the nominal level for all scenarios and both methods. The IP estimates of the baseline detection probability g_0 were close to the true value (0.1) and showed no trend with trap saturation (Table S1c).

IP estimates of density appeared slightly less precise than the ML estimates

across all levels of density and trap saturation. Curiously, the IP estimates reported by Efford (2004) appeared slightly more precise than either the present IP or present ML estimates. The inverse prediction estimator of Efford (2004) used different proxy variables (a null closed-population estimate $\hat{N}_0$ and the corresponding probability of capture $\hat{p}$, and the mean distance between successive capture locations of each individual) and simulations were performed in the DENSITY software (Efford, Dawson & Robbins 2004). Further checking would be needed to determine whether this is attributable to the differing proxies, or merely a random artifact (noting the 2004 study used only 100 replicates). The present estimates are otherwise consistent with those from 2004.

R code to run simulations

Each scenario is saved to separate file. In practice, scenarios are run in parallel on a computer cluster.

```
library(ipsecr)
library(secrdesign)
grid <- make.grid(12, 12, spacing = 30, detector = "single")
scen <- make.scenarios(D = c(0.5, 1, 2, 5, 10, 20, 50), g0 = 0.1,
    sigma = 40, noccasions = 5)
mask <- make.mask(grid, buffer = 200, type = "traprect")
nrepl <- 1000 # later reduced to 200 for IP with density >= 5/ha
for (task_id in 1:7) {
    # MLE
    sims1 <- run.scenarios(nrepl = nrepl, scen[task_id,],
        trapset = grid, xsigma = 5, maskset=mask, seed = 123,
        fit = TRUE, fit.function = "secr.fit", fit.args =
        list(ncores = 30, detectfn = "HN"))
    saveRDS(sims1, file = paste0("originalsimsML", task_id, ".RDS"))
    # IP
    sims2 <- run.scenarios(nrepl = nrepl, scen[task_id,],
        trapset = grid, xsigma = 5, maskset=mask, seed = 123,
        fit = TRUE, fit.function = "ipsecr.fit", fit.args =
        list(ncores = 15, detectfn = "HN"))
    saveRDS(sims2, file = paste0("originalsimsIP", task_id, ".RDS"))
}
```

Following code imports and tabulates results.

```
library(secrdesign)
sumone <- function (scenario = 1, prefix = "ML", parm = "D",
        stat = "RB", pm = 0.8) {
    sims <- readRDS(paste0("originalsims", prefix, scenario, ".RDS" ))
    st <- select.stats(sims, parameter = parm)
    # exclude estimates less than 20\% or greater than 180\% of true density
```

```
    permitted <- outer(st$scenarios[,parm], c(1-pm, 1+pm))
    st <- validate(st, "estimate", permitted, "all", quietly = TRUE)
    summary(st)$OUTPUT[[1]][stat,]
}

sumall <- function (pfix = "IP") {
    sumtab <- t(rbind(
    sapply(1:7, sumone, prefix = pfix, parm = "D", stat = "estimate"),
    sapply(1:7, sumone, prefix = pfix, parm = "D", stat = "RSE")[2:3,],
    sapply(1:7, sumone, prefix = pfix, parm = "D", stat = "RB")[2:3,],
    sapply(1:7, sumone, prefix = pfix, parm = "D", stat = "COV")[2,],
    sapply(1:7, sumone, prefix = pfix, parm = "g0", stat = "estimate")[2:3,],
    sapply(1:7, sumone, prefix = pfix, parm = "sigma", stat = "estimate")[2:3,]
    ))
    sumtab <- apply(sumtab, 2, as.numeric)
    sumtab[,4:8] <- 100*sumtab[,4:8]
    dimnames(sumtab)[[2]] <- c("repl", "Dhat", "seD", "RSE", "seRSE",
        "RB", "seRB", "COV", "g0hat", "seg0", "sigmahat", "sesigma")
    sumtab
}
sumall("ML") # Table S1b
sumall("IP") # Table S1c
```

References

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

Efford, M. G., Dawson, D. K., & Robbins C. S. (2004). DENSITY: software for analysing capture–recapture data from passive detector arrays. *Animal Biodiversity and Conservation*, **27,** 217–228.

Efford, M. G. (2022). secr: Spatially explicit capture-recapture models. R package version 4.5.8. https://CRAN.R-project.org/package=secr

Efford, M. G. (2023a). ipsecr: Spatially explicit capture-recapture using inverse prediction. R package version 1.4.0. https://CRAN.R-project.org/package=ipsecr

Efford, M. G. (2023b). secrdesign: Sampling design for spatially explicit capture-recapture. R package version 2.8.0. https://CRAN.R-project.org/package=secrdesign

Efford, M. G., Borchers, D. L., & Byrom, A. E. (2009). Density estimation by spatially explicit capture–recapture: likelihood-based methods. In *Modeling Demographic*

Processes in Marked Populations, D. L. Thomson, E. G. Cooch, & M. J. Conroy (eds), 255–269. New York: Springer.

Table S1: Comparative performance of IP and mis-specified ML estimators applied to simulated capture–recapture samples from populations differing in density D. Half-normal detection function $g_0 = 0.1$, $\sigma = 40$ m; 144 single-catch traps at 25-m spacing, 5 occasions. Mean (SE) of 1000 simulations (reduced to 200 for IP with D $\geq$ 5 ha^{-1}).

a. Proxy statistics n, p^* and σ^* (defined in main text) and trap saturation (TS)

D	n	p^*	σ^*	TS %
0.5	9.51 (0.10)	0.504 (0.003)	35.4 (0.18)	3.33 (0.04)
1	19.17 (0.14)	0.503 (0.002)	35.5 (0.11)	6.68 (0.05)
2	38.18 (0.20)	0.498 (0.001)	35.7 (0.08)	13.20 (0.07)
5	92.61 (0.30)	0.471 (0.001)	35.8 (0.06)	30.28 (0.10)
10	175.43 (0.38)	0.432 (0.000)	36.0 (0.04)	52.61 (0.11)
20	309.81 (0.42)	0.369 (0.000)	36.2 (0.04)	79.40 (0.08)
50	513.02 (0.34)	0.277 (0.000)	36.4 (0.04)	98.75 (0.01)

b. Maximum likelihood estimates using mis-specified model for multi-catch traps

D	$\hat{D}$	RSE($\hat{D}$) %	RB($\hat{D}$) %	COV($\hat{D}$) %	$\hat{g}_0$	$\hat{\sigma}$
0.5	0.49 (0.01)	37.2 (0.25)	–1.70 (1.00)	98.4	0.101 (0.001)	40.1 (0.21)
1	1.00 (0.01)	24.8 (0.10)	0.18 (0.77)	94.1	0.099 (0.001)	39.9 (0.14)
2	2.01 (0.01)	17.2 (0.05)	0.56 (0.53)	95.4	0.095 (0.001)	39.9 (0.10)
5	5.00 (0.02)	11.0 (0.02)	0.03 (0.33)	96.0	0.083 (0.000)	39.9 (0.06)
10	9.93 (0.03)	8.0 (0.01)	–0.69 (0.26)	94.5	0.069 (0.000)	39.9 (0.05)
20	19.86 (0.04)	6.3 (0.00)	–0.71 (0.21)	93.1	0.048 (0.000)	40.0 (0.05)
50	49.95 (0.10)	6.3 (0.01)	–0.11 (0.20)	94.2	0.023 (0.000)	39.7 (0.05)

c. Estimates by simulation and inverse prediction

D	$\hat{D}$	RSE($\hat{D}$) %	RB($\hat{D}$) %	COV($\hat{D}$) %	$\hat{g}_0$	$\hat{\sigma}$
0.5	0.52 (0.01)	43.0 (1.05)	3.76 (1.00)	97.6	0.098 (0.001)	41.3 (0.23)
1	1.03 (0.01)	26.6 (0.18)	2.81 (0.78)	95.2	0.098 (0.001)	40.7 (0.14)
2	2.03 (0.01)	18.0 (0.07)	1.34 (0.54)	96.3	0.098 (0.006)	40.5 (0.11)
5	5.03 (0.04)	11.3 (0.06)	0.58 (0.77)	95.5	0.098 (0.001)	40.3 (0.16)
10	10.00 (0.06)	8.1 (0.04)	0.02 (0.61)	92.5	0.097 (0.001)	40.3 (0.11)
20	20.15 (0.09)	6.4 (0.03)	0.76 (0.48)	93.5	0.098 (0.001)	40.1 (0.11)
50	50.44 (0.23)	6.5 (0.04)	0.89 (0.46)	96.0	0.101 (0.001)	40.2 (0.12)

Appendix 2: Brushtail possum examples

The examples use data included in the package 'secr' (Efford, 2022). Packages 'secr' (≥4.5.7), 'ipsecr' (Efford 2023a) and 'sf' are required. The hazard half-normal detection function ('HHN') is defined by $\lambda(r) = \lambda_0 \exp(-r^2/(2\sigma^2))$. The habitat mask `msk` is a grid of points at 7.5 m spacing within 120 m of a trap, excluding areas of shingle riverbed delineated in the `boundary` file. 'lcl' and 'ucl' are 95% confidence limits.

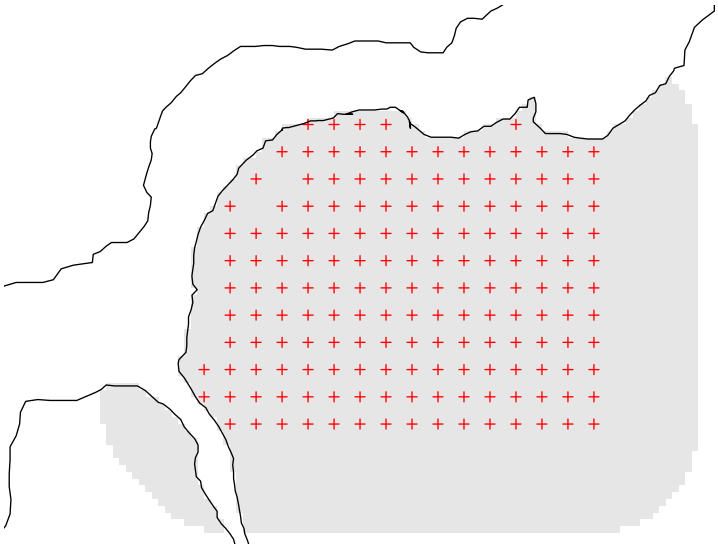

Figure S1: Brushtail possum trapping grid 30-m spacing (Efford & Cowan 2004). Shaded area indicates extent of habitat mask.

Example 1: Brushtail possums, February 1996

```
#----------------------------------------------------------------
library(ipsecr)
set.seed(123)
setNumThreads(7) # number of cores for parallel processing
Feb96 <- OVpossumCH[[1]] # one trapping session (February 1996)
#----------------------------------------------------------------
# prepare habitat mask, excluding riverbed
boundary <- system.file("extdata/OVforest.shp", package = "secr")
OVforest <- sf::st_read(boundary)
msk <- make.mask(traps(Feb96), buffer = 120, type = "trapbuffer",
    poly = OVforest[1:2,], spacing = 7.5, keep.poly = FALSE)
#----------------------------------------------------------------
```

```
# display
plot(traps(Feb96), border = 120, gridlines = FALSE)
plot(msk, dots = FALSE, add = TRUE, col = grey(0.9))
plot(traps(Feb96), add = TRUE)
plot(sf::st_union(OVforest), add = TRUE)
#-----------------------------------------------------------------
# mis-specified multi-catch model, MLE
fitML <- secr.fit(Feb96, mask = msk, detectfn = "HHN", trace = F)
predict(fitML)
# link estimate SE.estimate lcl ucl
# D log 14.3802 1.003093 12.5447 16.4842
# lambda0 log 0.1015 0.008948 0.0854 0.1206
# sigma log 27.3765 0.972969 25.5349 29.3508
#-----------------------------------------------------------------
# single-catch model, simulation and inverse prediction
fitIP <- ipsecr.fit(Feb96, mask = msk, detectfn = "HHN", verbose = T)
predict(fitIP)
# link estimate SE.estimate lcl ucl
# D log 14.1773 0.90293 12.5152 16.060
# lambda0 log 0.1555 0.01552 0.1279 0.189
# sigma log 27.4818 0.89938 25.7748 29.302
#-----------------------------------------------------------------
# mis-specified multi-catch model, simulation and inverse prediction
fitIPm <- ipsecr.fit(Feb96, mask = msk, detectfn = "HHN",
    details = list(newdetector = "multi"), verbose = FALSE)
predict(fitIPm)
# link estimate SE.estimate lcl ucl
# D log 14.2402 1.002276 12.40734 16.3438
# lambda0 log 0.1034 0.009787 0.08589 0.1244
# sigma log 27.5175 1.036710 25.55944 29.6255
#-----------------------------------------------------------------
```

3-D plot

The 3-D plot in the main text can be reproduced with this code:

```
library(plot3D) # may require installation
# fit model as before, but keep all simulations
fitIPk <- ipsecr.fit(Feb96, mask = msk, detectfn = "HHN", verbose = FALSE, details =
    list(keep.sim = TRUE))
par(mfrow=c(2,2), oma = c(1,1,3,1), mar = c(2,2,0.5,0), lwd=1, cex=1)
oldplot <- plot3D.IP(fitIPk, box = 1)
plot3D.IP(fitIPk, box = 2, oldplot, plotfinal = TRUE)
mtext(outer =TRUE, side = 3, line = 0.5, adj = c(0.2,0.8), cex = 1.1,
    c("Parameter space", "Proxy space"))
```

Example 2: Brushtail possums, February 1996 and February 1997, with interference

In addition to the preceding data from package 'secr', this requires the text files 'NTFeb1996.txt' and 'NTFeb1997.txt' that are available on Zenodo (Efford 2023b). Output shown as comments (#) has been slightly edited for formating.

```
#----------------------------------------------------------------
library(ipsecr)
set.seed(123)
options(digits = 4)
setNumThreads(7) # number of cores for parallel processing
#----------------------------------------------------------------
# extract two sessions and read non-target data
Feb9697 <- subset(OVpossumCH, sessions = c(1,4))
names(Feb9697) <- 1996:1997
nontarget(Feb9697) <- lapply(c("NTFeb1996.txt", "NTFeb1997.txt"),
    read.table, head = TRUE, row.names = 1)
#----------------------------------------------------------------
# prepare habitat mask, excluding riverbed, as in Example 1
boundary <- system.file("extdata/OVforest.shp", package = "secr")
OVforest <- sf::st_read(boundary)
msk <- make.mask(traps(Feb9697), buffer = 120, type = "trapbuffer",
    poly = OVforest[1:2,], spacing = 7.5, keep.poly = FALSE)
#----------------------------------------------------------------
# mis-specified multi-catch model, MLE, constant lambda0
fitMLs <- secr.fit(Feb9697, model = list(D~session), mask = msk, detectfn = "HHN",
    trace = FALSE)
predict(fitMLs)
#‘session = 1996‘
# link estimate SE.estimate lcl ucl
# D log 13.42246 0.914274 11.7468 15.337
# lambda0 log 0.09283 0.005823 0.0821 0.105
# sigma log 31.56650 0.808716 30.0208 33.192
# ‘session = 1997‘
# link estimate SE.estimate lcl ucl
# D log 9.75086 0.776283 8.3442 11.395
# lambda0 log 0.09283 0.005823 0.0821 0.105
# sigma log 31.56650 0.808716 30.0208 33.192
#----------------------------------------------------------------
# mis-specified multi-catch model, MLE, year-specific lambda0
fitMLsls <- secr.fit(Feb9697, model = list(D~session, lambda0~session), mask = msk,
    detectfn = "HHN", trace = FALSE)
predict(fitMLsls)
# ‘session = 1996‘
# link estimate SE.estimate lcl ucl
# D log 14.06482 0.981481 12.26895 16.12357
# lambda0 log 0.07872 0.005986 0.06784 0.09135
# sigma log 31.56622 0.809195 30.01966 33.19246
```

```
# ‘session = 1997‘
# link estimate SE.estimate lcl ucl
# D log 9.2916 0.741698 7.94793 10.8625
# lambda0 log 0.1147 0.008852 0.09863 0.1334
# sigma log 31.5662 0.809195 30.01966 33.1925
#-----------------------------------------------------------------
# inverse prediction, ignoring interference
fitIPs <- ipsecr.fit(Feb9697, model = list(D~session), mask = msk, detectfn = "HHN",
    verbose = FALSE, details = list(ignorenontarget = TRUE)
predict(fitIPs)
# ‘session = 1996‘
# link estimate SE.estimate lcl ucl
# D log 12.8816 0.961604 11.1305 14.9081
# lambda0 log 0.1352 0.008496 0.1196 0.1529
# sigma log 31.5914 0.831311 30.0036 33.2632
# ‘session = 1997‘
# link estimate SE.estimate lcl ucl
# D log 10.3525 0.747267 8.9884 11.9236
# lambda0 log 0.1352 0.008496 0.1196 0.1529
# sigma log 31.5914 0.831311 30.0036 33.2632
#-----------------------------------------------------------------
# inverse prediction, constant interference
fitIPsi <- ipsecr.fit(Feb9697, model = list(D~session, NT~1), mask = msk, detectfn = "
    HHN", verbose = FALSE)
predict(fitIPsi)
# ‘session = 1996‘
# link estimate SE.estimate lcl ucl
# D log 13.0838 0.99784 11.2696 15.190
# lambda0 log 0.2447 0.01688 0.2137 0.280
# sigma log 31.5097 0.84188 29.9024 33.203
# NT log 0.9586 0.04175 0.8802 1.044
# ‘session = 1997‘
# link estimate SE.estimate lcl ucl
# D log 10.1994 0.72858 8.8684 11.730
# lambda0 log 0.2447 0.01688 0.2137 0.280
# sigma log 31.5097 0.84188 29.9024 33.203
# NT log 0.9586 0.04175 0.8802 1.044
#-----------------------------------------------------------------
# inverse prediction, session-specific interference
fitIPsis <- ipsecr.fit(Feb9697, model = list(D~session, NT~session), mask = msk,
    details = list(max.nsim = 20000), detectfn = "HHN", verbose = FALSE)
predict(fitIPsis)
# ‘session = 49‘
# link estimate SE.estimate lcl ucl
# D log 14.9621 1.10522 12.9480 17.2896
# lambda0 log 0.2718 0.01832 0.2382 0.3102
# sigma log 31.5555 0.77649 30.0699 33.1144
# NT log 1.4987 0.10041 1.3145 1.7088
# ‘session = 52‘
# link estimate SE.estimate lcl ucl
# D log 8.6502 0.48481 7.7510 9.6537
# lambda0 log 0.2718 0.01832 0.2382 0.3102
```

```
# sigma log 31.5555 0.77649 30.0699 33.1144
# NT log 0.7118 0.05541 0.6112 0.8289
#------------------------------------------------------------------
```

Annual change in density under the various models is shown in Fig. S2.

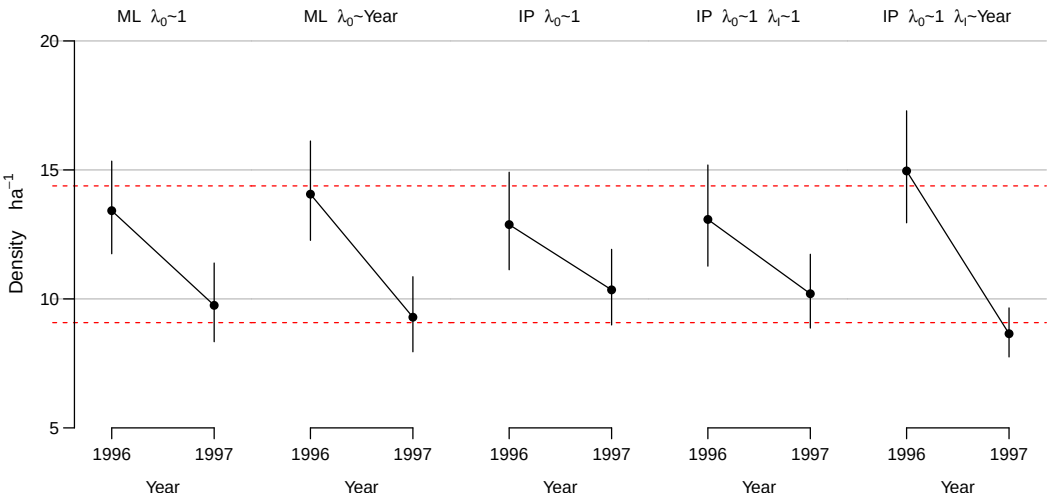

Figure S2: Change in brushtail possum density between February 1996 and February 1997 from 5 fitted models. The last two models include non-target interference (λ_I). 95% CI. Dashed lines indicate 1996 (upper) and 1997 (lower) estimates from fully year-specific ML models.

Example 2a: Varying spatial scale of detection

This example allows the spatial scale of detection to differ spatially, specifically between forest types. The `proxy.ms` function models the individual-level RPSV as a function of forest type at the centroid of each individual's detections. It requires a spatial data source for the forest type, at least for points within the outer hull of the detector array. This is given in the '...' argument: a named component 'spatialdata' is passed by `proxy.ms` to its internal call of the `addCovariates` function. Here we use a habitat mask with 'forest' as a covariate (coincidentally the same one used as the mask argument of `ipsecr.fit`).

```
#------------------------------------------------------------------
library(ipsecr)
set.seed(123)
setNumThreads(7) # number of cores for parallel processing
Feb96 <- OVpossumCH[[1]] # one trapping session (February 1996)
#------------------------------------------------------------------
# prepare habitat mask, excluding riverbed
```

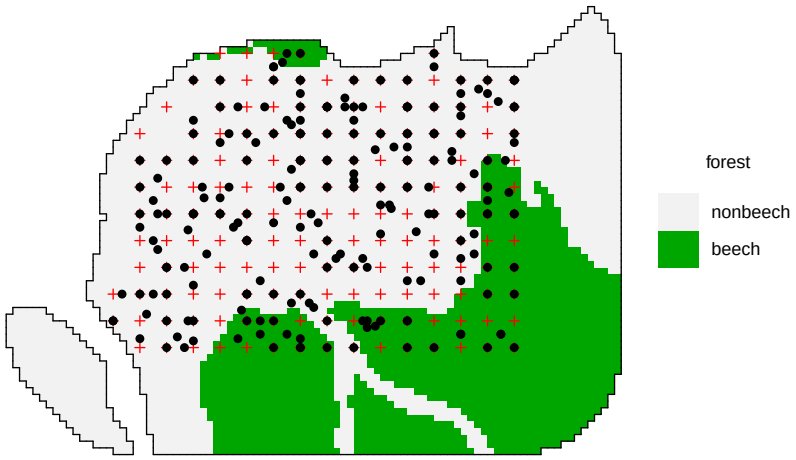

Figure S3: Centroids of brushtail possum capture locations in relation to forest type. February 1996.

```
boundary <- system.file("extdata/OVforest.shp", package = "secr")
OVforest <- sf::st_read(boundary)
msk <- make.mask(traps(Feb96), buffer = 120, type = "trapbuffer",
    poly = OVforest[1:2,], spacing = 7.5, keep.poly = FALSE)
#-----------------------------------------------------------------
# add "forest" covariate to mask and plot
msk <- addCovariates(msk, OVforest[1:2,])
plot(msk, cov = "forest", dots = FALSE, legend = TRUE)
plotMaskEdge(msk, add = TRUE)
plot(traps(Feb96), add = TRUE)
points(centroids(Feb96), pch = 16)
#-----------------------------------------------------------------

fit <- ipsecr.fit(Feb96, mask = msk, model = sigma~forest,
    spatialdata = msk)

predict(fit, all.levels = TRUE)

# 'forest = beech'
# link estimate SE.estimate lcl ucl
# D log 14.0760338 0.9693785 12.3006956 16.1076034
# g0 logit 0.1560139 0.0185713 0.1229674 0.1959572
# sigma log 30.7520023 4.9417092 22.4884760 42.0520112

# 'forest = nonbeech'
# link estimate SE.estimate lcl ucl
```

```
# D log 14.0760338 0.9693785 12.3006956 16.1076034
# g0 logit 0.1560139 0.0185713 0.1229674 0.1959572
# sigma log 26.0724766 1.8208569 22.7409253 29.8921010
```

It appears that the scale of detection is smaller in the mixed non-beech forest than in the beech (*Nothofagus*) forest. This agrees with the analysis of Efford et al. (2016) who found that the scale of detection varied inversely with population density, which was lower in beech forest. However, there is relatively little beech forest within the trapped area, and the corresponding $\hat{\sigma}$ is rather imprecise (Fig. S3).

Discussion

An independent trap-catch index of ship rat density declined from 8.5 (SE) in February 1996 to 0.6 (SE) in February 1997 (Efford et al. 2006).

References

Efford, M. G., Dawson, D. K., Jhala, Y. V. & Qureshi, Q. (2016). Density-dependent home range size revealed by spatially explicit capture–recapture. *Ecography*, **39,** 676–688.

Efford, M. G. (2022). secr: Spatially explicit capture-recapture models. R package version 4.5.8. https://CRAN.R-project.org/package=secr

Efford, M. G. (2023a). ipsecr: Spatially explicit capture-recapture using inverse prediction. R package version 1.4.0. https://CRAN.R-project.org/package=ipsecr

Efford, M. G. (2023b). Spatially explicit capture–recapture by inverse prediction. *Zenodo*, https://doi.org/10.5281/zenodo.7668378

Efford, M. G. & Cowan, P. E. (2004). Long-term population trend of *Trichosurus vulpecula* in the Orongorongo Valley, New Zealand. In R. L. Goldingay & S. M. Jackson (Eds.) *The Biology of Australian Possums and Gliders* (pp. 471–483). Surrey Beatty & Sons.

Efford, M. G., Fitzgerald, B. M., Karl, B. J. & Berben, P. H. (2006). Population dynamics of the ship rat *Rattus rattus* L. in the Orongorongo Valley, New Zealand.

New Zealand Journal of Zoology, **33,** 273–297.

Appendix 3: Simulations with density trend and single-catch traps

If traps fill up, varying trap saturation may be confounded with varying density in a naive spatially explicit capture–recapture model (Distiller & Borchers 2015; main text). Simulation and inverse prediction in 'ipsecr' enables a correctly specified model to be fitted to data single-catch traps, and consequently enables almost unbiased estimation of density trend, independent of trap saturation.

This is illustrated by simulation of a scenario in which density follows a log-linear trend in the east-west direction. Similar scenarios were considered by Efford et al. (2009) and Distiller & Borchers (2015).

Single-catch traps were placed on a square grid at notional spacing 20 m within a notional habitat extending 100 m beyond the traps in the cardinal directions. The population followed an inhomogeneous Poisson distribution with intensity (expected density) $D(x) = \exp(\beta_0 + \beta_1 x_s)$ where $\beta_0 = 3$, $\beta_1 = 1$ and x_s indicates the value of x scaled to zero mean and unit standard deviation over the extent of the notional habitat. This corresponds to a density of 9.9 per hectare at the western edge of the grid and 41.9 per hectare at the eastern edge, and 31.7 per hectare averaged over the entire habitat.

The detection function used for both data generation and model fitting was half-normal on the hazard scale, with parameters $\lambda_0 = 0.2$ and $\sigma = 25$. In a real population we might expect σ to vary inversely with density (Efford et al. 2016); a simple scenario with constant σ is used here merely to illustrate the ipsecr approach.

```
library(ipsecr) # also loads secr

# make 8x8 grid of single-catch traps
tr <- make.grid(8, 8, spacing = 20, detector = "single")

# define discretized habitat (default 64 x 64 pixels)
msk <- make.mask(tr, buffer = 100)

# covariate "Dexp" is inhomogeneous Poisson intensity for sim.pop
# east-west trend linear on log scale with slope 1 and intercept 3
covariates(msk) <- data.frame(x0 = scale(msk$x),
    Dexp = exp(scale(msk$x) + 3))
```

```
# other setup
fitsIP <- list()
fitsML <- list()
setNumThreads(7)
set.seed(1237)

for (r in 1:100) {
    # simulate population and sample over 5 occasions
    popexp <- sim.popn(D = "Dexp", core = msk, model2D = "IHP", buffer = 100)
    chexp <- sim.capthist(tr, popn = popexp, detectfn = "HHN",
        noccasions = 5, detectpar = list(lambda0 = 0.2, sigma = 25))
    # fit model by simulation and inverse prediction
    fitsIP[[r]] <- ipsecr.fit(chexp, mask = msk, detectfn = "HHN", model = list(D~x))
    # fit mis-specified model (using likelihood for multi-catch traps)
    fitsML[[r]] <- secr.fit(chexp, mask = msk, detectfn = "HHN", model = list(D~x0))
}

# extract coefficients (100 x 2 matrix)
outIP <- t(sapply(fitsIP, function(x) coef(x)[1:2,1]))
outML <- t(sapply(fitsML, function(x) coef(x)[1:2,1]))

# function to add curve with coefficients a,b on log scale
plotone <- function(ab, x, ...) {
    x0 <- scale(x, mean(-100:240), sd(-100:240))
    lines(x, exp(x0*ab[2]+ab[1]), ...)
}

# x values for plotting
x <- -100:240
par(mfrow=c(1,1), mgp=c(2.2,0.6,0), las = 1)
plot(0,0, type = "n", xlim = c(-100,240), ylim = c(0,100),
    xlab = "x", ylab = "D(x)")

# each replicate ipsecr.fit
apply(out,1, plotone, x = x, lwd = 1,col = "grey")

# true trend
plotone(c(3,1), x = x, type = "l", lwd = 2, col = "black")

# average of simulations, ipsecr.fit
plotone(apply(out,2,mean), x = x, lty = 2, lwd = 2.5, col = "blue")

# average of simulations, mis-specified (multi-catch)
plotone(apply(outMC,2,mean), x = seq(-100,240,4), pch = 16,
    cex = 0.5, type = "p", col = "red")

# limits of grid
abline(v=c(0,140), lty=2)
rug(unique(tr$x), col="red", lwd=2)
```

References

Distiller, G. & Borchers, D. L. (2015). A spatially explicit capture–recapture estimator for single-catch traps. *Ecology and Evolution* **5,** 5075–5087.

Efford, M. G., Borchers, D. L., & Byrom, A. E. (2009). Density estimation by spatially explicit capture–recapture: likelihood-based methods. In *Modeling Demographic Processes in Marked Populations*, D. L. Thomson, E. G. Cooch, & M. J. Conroy (eds), 255–269. New York: Springer.

Efford, M. G., Dawson, D. K., Jhala, Y. V. & Qureshi, Q. (2016). Density-dependent home range size revealed by spatially explicit capture–recapture. *Ecography*, **39,** 676–688.

Appendix 4: Simulations of post-collection subsampling

Some sampling protocols yield more samples at each detector than can be processed, and it is common for some sort of post-collection subsampling to be undertaken. This applies specifically to the use of hairs for DNA individuation (e.g., Jiménez et al., 2021). In the extreme, a single hair sample may be chosen for laboratory analysis from among those left at a hair snag. Individual detection probability then declines with density, and detections violate the usual assumption of independence among animals. The effect is to bias estimates of λ_0 if a binary proximity model is fitted (independent detections, maximum one per individual per detector per occasion).

'secr' (Efford, 2022) has the 'capped' detector type, in which a maximum of one detection may occur at a detector on any occasion, but the maximum likelihood method for capped detectors in 'secr.fit' is an unpublished approximation. Simulation and inverse prediction is one way to tackle the problem (see also Jiménez et al., 2021). 'ipsecr.fit' recognises the 'capped' detector type and simulates accordingly (using simCH); the default proxies apply. A call such as this is sufficient if traps(ch) has detector type 'capped':

```
ipsecr.fit(ch)
```

R package 'secrdesign' (Efford 2023a) has recently been extended to allow models to be fitted with the function 'ipsecr.fit' from 'ipsecr' (Efford 2023b). This makes it easy to examine the performance of the method across multiple scenarios. Example code follows. Fig. S4 compares results from the ipsecr estimator with naive estimates from a binary proximity model, across five levels of density (0.0625–2 / σ^2), and consequent detector saturation, and two levels of λ_0 (with compensatory adjustment of σ). The naive model shows large bias in $\hat{\lambda}_0$ whenever there is sigificant detector saturation. Bias in the ipsecr estimates is positive and slight, except at the lowest density. The observed low-density bias is possibly due to the very small samples and failure of a significant minority of simulations (3.9% when $\lambda_0 = 0.2$; 2.0% when $\lambda_0 = 0.05$, none otherwise).

```
library(secrdesign) # >= 2.8.0

# define grid of "capped" detectors
```

```
# (proximity detectors subject to post-collection sampling)
trc <- make.grid(10,10, spacing = 20, detector = "capped")

# generate data.frame of simulation scenarios
scenc <- make.scenarios (
    D = c(1,4,8,16,32) / (20/100)^2 / 16,
    lambda0 = c(0.2,0.05),
    sigma = 20,
    detectfn = "HHN",
    noccasions = 5
)
scenc$sigma[scenc$lambda0 == 0.05] <- 40
scenc$maskindex <- rep(1:2, each = 5)

# vary extent of habitat for each scale of detection
masks <- list(
    make.mask(trc, buffer = 4 * 20, spacing = 10, type = "trapbuffer"),
    make.mask(trc, buffer = 4 * 40, spacing = 10, type = "trapbuffer")
)

# run.scenarios internally generates data with the secr functions
# sim.popn and sim.capthist as in Example 3

simc <- run.scenarios(
    nrepl = 2,
    scenarios = scenc,
    trapset = trc,
    maskset = masks,
    ncores = 6,
    seed = 1234,
    fit = TRUE,
    fit.function = "ipsecr.fit",
    fit.args = list(detectfn = "HHN", verbose = FALSE))

# overall summary focussing on density (D)
summary(simc)

# summarize estimate of lambda0, filtering for bad fits at lowest density
st <- select.stats(simc, "lambda0")
st <- validate(st, "SE.estimate", c(1e-3, 0.2), "all")
summary(st)
```

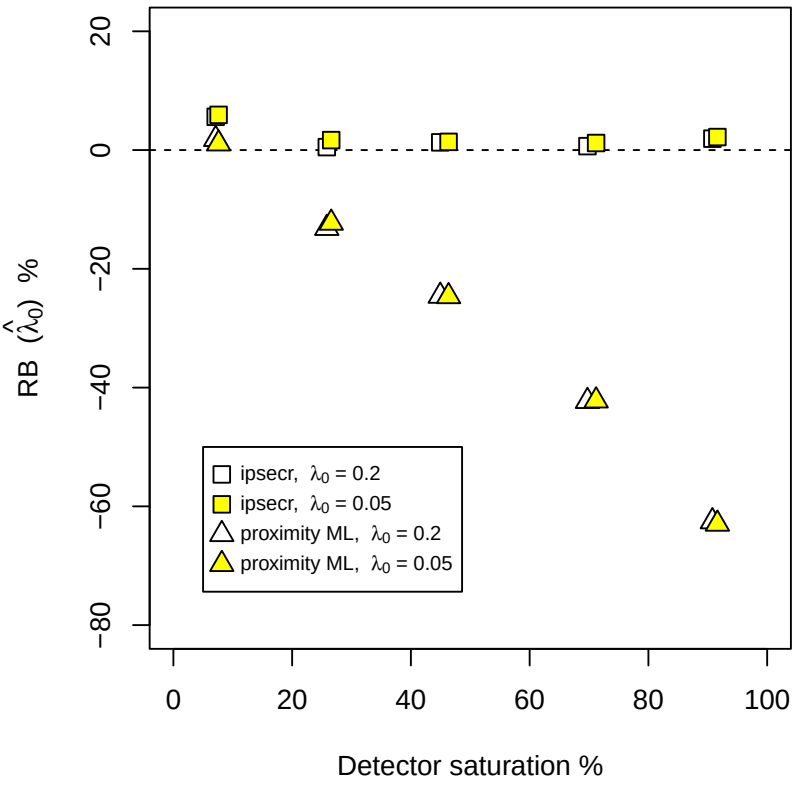

Figure S4: Simulations of post-collection subsampling: bias in estimate of detection intercept λ_0.

References

Efford, M. G. (2022). secr: Spatially explicit capture-recapture models. R package version 4.5.8. https://CRAN.R-project.org/package=secr

Efford, M. G. (2023a). secrdesign: Sampling design for spatially explicit capture–recapture. R package version 2.8.0. https://CRAN.R-project.org/package=secrdesign and https://github.com/MurrayEfford/secrdesign

Efford, M. G. (2023b). ipsecr: Spatially explicit capture-recapture using inverse prediction. R package version 1.4.0. https://CRAN.R-project.org/package=ipsecr

Jiménez, J., Augustine, B. C., Linden, D. W., Chandler, R. B., and Royle, J. A. (2021). Spatial capture–recapture with random thinning for unidentified encounters. *Ecology and Evolution* **11,** 1187–1198.