Simple SCR exercises

The code in the file `scr-ll.r` creates a log-likelihood function, loads some example data, and fits the 'binary proximity model' described by Efford et al. (2009). The model makes the following assumptions:

- The number of animals' activity centres in the survey region is a Poisson random variable, with expectation equal to animal density, D, multiplied by the area of the survey region.
- The activity centre locations are independent, and are uniformly distributed across the survey region.
- The probability that a detector detects an individual is given by the halfnormal detection function,

  $$g(d) = g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right),$$

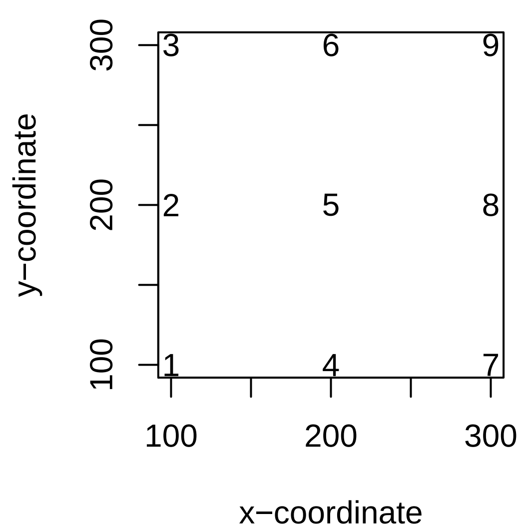
  where d is the distance between the animal's activity centre and the detector.

Run the code in `scr-ll.r` and answer the following questions.

General questions

1. Create a plot of the detector locations. Their coordinates can be found in `test.data$traps`.

```r
par(mar = c(4, 4, 0, 0), oma = rep(1, 4))
plot(test.data$traps, asp = 1, pch = as.character(1:9),
     xlab = "x-coordinate", ylab = "y-coordinate")
```



We have a three-by-three grid of detectors with a spacing of 100 m between them.

2. Inspect the capture histories in `test.data$bin.capt`. Describe what the first two rows represent.

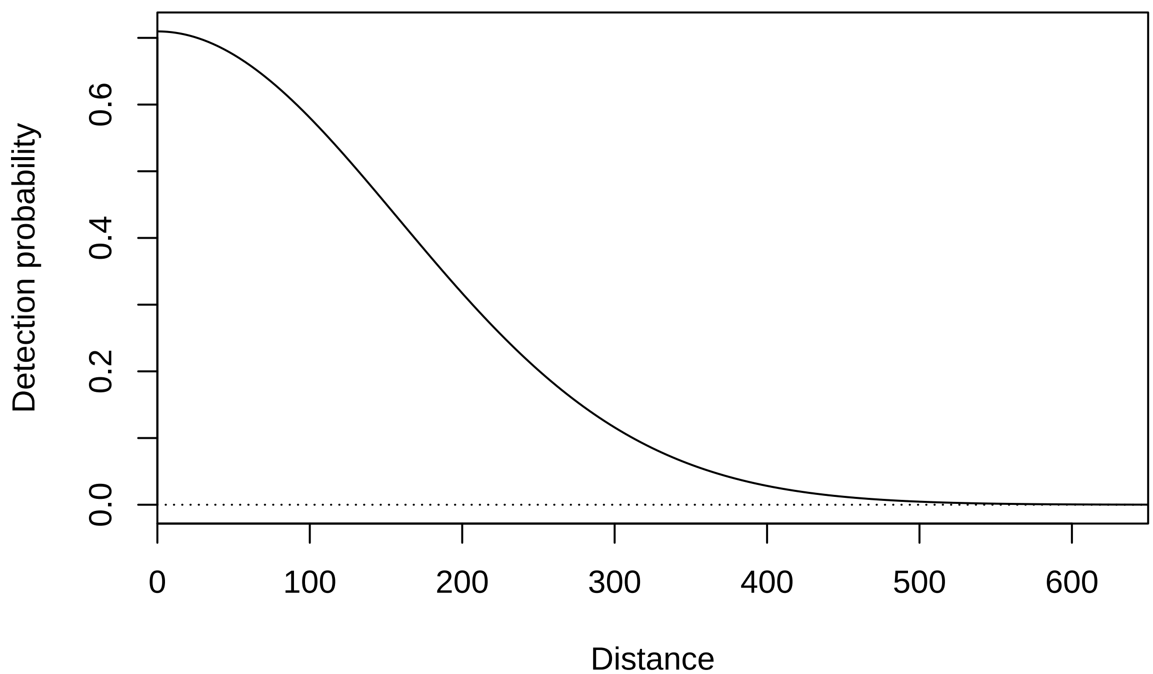
```
head(test.data$bin.capt, 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    0    0    0    1    0    0    0    0
## [2,]    1    1    0    0    1    0    0    1    1
```

The first animal was detected by detectors 1 and 5 (bottom-left and middle). The second animal was detected by detectors 1, 2, 5, 8, and 9 (bottom-left, middle-left, middle, middle-right, and top-right).

3. The model has estimated animal density, D, and parameters of a halfnormal detection function, g_0 and σ. Create a plot of the detection function estimated by the model.

```
par(mar = c(4, 4, 0, 0), oma = rep(0.1, 4), xaxs = "i")
xx <- seq(0, 650, length.out = 1000)
g0 <- plogis(fit$par[2])
sigma <- exp(fit$par[3])
yy <- g0*exp(-xx^2/(2*sigma^2))
plot(xx, yy, type = "l", xlab = "Distance", ylab = "Detection probability")
abline(h = 0, lty = "dotted")
```



4. Tricky question for STATS 730 graduates only:

   (a) Compute standard errors for the transformed parameters, $\log(D)$, $\text{logit}(g_0)$, and $\log(\sigma)$. The `optim()` argument `hessian` or the `optimHess()` function will be useful.

   ```
   hess <- optimHess(fit$par, scr.nll, capt = test.data$bin.capt,
                     traps = test.data$traps, mask = test.data$mask)
   vcov.link <- solve(hess)
   sqrt(diag(vcov.link))
   ```

   ```
   ## [1] 0.7539109 0.7682073 0.4346113
   ```

   (b) Compute standard errors for the parameters themselves, D, g_0, and σ.

```r
jacobian <- diag(3)
diag(jacobian) <- c(exp(fit$par[1]), dlogis(fit$par[2]), exp(fit$par[3]))
vcov.unlink <- jacobian %*% vcov.link %*% t(jacobian)
sqrt(diag(vcov.unlink))

## [1]  0.2327907  0.1582687 68.4868909
```

(c) Compute confidence intervals for the three parameters.

Calculating a confidence interval for the transformed parameters and then back-transforming.

```r
ses.link <- sqrt(diag(vcov.link))
## For D:
exp(fit$par[1] + c(-1, 1)*qnorm(0.975)*ses.link[1])

## [1] 0.07045558 1.35324262

## For g0:
plogis(fit$par[2] + c(-1, 1)*qnorm(0.975)*ses.link[2])

## [1] 0.3516708 0.9167979

## For sigma:
exp(fit$par[3] + c(-1, 1)*qnorm(0.975)*ses.link[3])

## [1]  67.23024 369.35863
```

Constructing the confidence intervals on the untransformed parameters directly is possible, but not recommended. We would expect the transformed versions to be better approximated by a normal distribution. We also get confidence intervals going out-of-bounds (negative $D < 0$ and $g_0 > 1$) if we use this approach.

```r
ses.unlink <- sqrt(diag(vcov.unlink))
## For D:
exp(fit$par[1]) + c(-1, 1)*qnorm(0.975)*ses.unlink[1]

## [1] -0.1474839  0.7650388

## For g0:
plogis(fit$par[2]) + c(-1, 1)*qnorm(0.975)*ses.unlink[2]

## [1] 0.3995048 1.0199068

## For sigma:
exp(fit$par[3]) + c(-1, 1)*qnorm(0.975)*ses.unlink[3]

## [1]  23.35011 291.81378
```

Bonus points if you computed profile-likelihood confidence intervals—that's probably the best option, but quite a lot more work!

5. Write some R code that simulates capture histories from a spatial capture-recapture model under the following conditions:

- The survey region is a square, with x-coordinate limits $(-500, 900)$ and y-coordinate limits also $(-500, 900)$. Note that these coordinates are given in metres.
- Detectors are deployed on a three-by-three grid with a 100 m spacing between them, so that the columns are located at x-coordinates 100, 200, and 300, and the rows are located at y-coordinates 100, 200, and 300. Note that this is the configuration of the detectors in `test.data$traps`.
- Animal density is $D = 0.75$ animals per hectare. Note that 1 hectare is $10\,000$ m^2.
- Conditional on its activity centre location, an individual is detected by a detector with

probability given by a halfnormal detection function with $g_0 = 0.9$ and $\sigma = 75$ m.

For bonus points, write your R code as a function, allowing the user to set their own detector locations and parameter values.

```
## Arguments:
##
## pars: A vector of parameters (D, g0, sigma).
## region.lims: A vector of survey region limits (x.lower, x.upper, y.lower, y.upper).
## traps: Detector locations as a matrix of coordinates.
sim.simplescr <- function(pars, region.lims, traps){
    ## Extracting parameter values.
    D <- pars[1]
    g0 <- pars[2]
    sigma <- pars[3]
    ## Calculating area of survey region in hectares.
    region.area <- (region.lims[2] - region.lims[1])*
        (region.lims[4] - region.lims[3])/10000
    ## Extracting number of detectors.
    n.traps <- nrow(traps)
    ## Simulating number of animals.
    n.acs <- rpois(1, D*region.area)
    ## Simulating activity centre locations.
    ac.locs.x <- runif(n.acs, region.lims[1], region.lims[2])
    ac.locs.y <- runif(n.acs, region.lims[3], region.lims[4])
    ac.locs <- cbind(ac.locs.x, ac.locs.y)
    ## Calculating a matrix of differences between activity centres and detectors.
    dists <- crossdist(ac.locs[, 1], ac.locs[, 2],
                       traps[, 1], traps[, 2])
    ## Calculating detection probabilities.
    det.probs <- g0*exp(-dists^2/(2*sigma^2))
    ## Creating capture histories.
    capt.full <- matrix(rbinom(n.acs*n.traps, 1, det.probs),
                        nrow = n.acs, ncol = n.traps)
    ## We only observe capture histories with at least one detection.
    capt <- capt.full[apply(capt.full, 1, sum) > 0, ]
    capt
}
set.seed(1234)
capt.sim <- sim.simplescr(c(0.75, 0.9, 75), c(-500, 900, -500, 900),
                          test.data$traps)
capt.sim
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]    0    0    0    0    0    0    1    1    0
##  [2,]    0    0    0    1    1    0    0    1    0
##  [3,]    0    0    0    1    0    0    1    0    0
##  [4,]    1    0    0    0    0    0    0    0    0
##  [5,]    0    0    1    0    0    0    0    0    0
##  [6,]    0    0    0    0    0    0    1    1    0
##  [7,]    0    1    1    0    0    1    0    1    0
##  [8,]    0    0    0    0    0    1    0    0    1
##  [9,]    0    0    0    0    0    1    0    0    0
## [10,]    0    0    0    1    0    0    0    1    1
## [11,]    0    0    0    0    1    0    0    0    0
## [12,]    0    0    1    0    0    0    0    0    0
## [13,]    0    0    0    1    0    0    0    0    0
## [14,]    1    1    0    1    0    0    0    0    0
```

```
## [15,]    0    1    0    0    1    0    1    1    0
```

6. Fit a spatial capture-recapture model to your simulated data. Note that you can use the detector locations in `test.data$traps` and the mask in `test.data$mask`. How close are your estimates to the true parameter values?

```r
fit.sim <- optim(par.start, scr.nll, capt = capt.sim, traps = test.data$traps,
                 mask = test.data$mask)
## For D:
exp(fit.sim$par[1])
```

```
## [1] 0.7699099
```

```r
## For g0:
plogis(fit.sim$par[2])
```

```
## [1] 0.9182916
```

```r
## For sigma:
exp(fit.sim$par[3])
```

```
## [1] 85.10117
```

All three are surprisingly close. In fact, closer than I'd expect—I think this is partly luck!

7. For STATS 730 graduates only: Compute confidence intervals for the three parameters. Did they capture the true parameter values?

```r
hess.sim <- optimHess(fit.sim$par, scr.nll, capt = capt.sim,
                      traps = test.data$traps, mask = test.data$mask)
vcov.link.sim <- solve(hess.sim)
ses.link.sim <- sqrt(diag(vcov.link.sim))
## For D:
exp(fit.sim$par[1] + c(-1, 1)*qnorm(0.975)*ses.link.sim[1])
```

```
## [1] 0.3905592 1.5177245
```

```r
## For g0:
plogis(fit.sim$par[2] + c(-1, 1)*qnorm(0.975)*ses.link.sim[2])
```

```
## [1] 0.03736101 0.99969282
```

```r
## For sigma:
exp(fit.sim$par[3] + c(-1, 1)*qnorm(0.975)*ses.link.sim[3])
```

```
## [1]  57.62621 125.67560
```

Yes, all three parameters fall within their respective confidence intervals.

8. Run a simulation study, repeating Questions 5–7 a total of 100 times. This involves simulating 100 sets of capture histories, and generating estimates from each. Inspect your 100 sets of estimates.

```r
set.seed(4321)
n.sims <- 100
par.ests <- matrix(0, nrow = n.sims, ncol = 3)
D.cis <- matrix(0, nrow = n.sims, ncol = 2)
g0.cis <- matrix(0, nrow = n.sims, ncol = 2)
sigma.cis <- matrix(0, nrow = n.sims, ncol = 2)
for (i in 1:n.sims){
    cat(i, "\n")
    capt.sim <- sim.simplescr(c(0.75, 0.9, 75), c(-500, 900, -500, 900),
                              test.data$traps)
    fit.sim <- optim(par.start, scr.nll, capt = capt.sim, traps = test.data$traps,
```

```r
                     mask = test.data$mask)
    par.ests[i, ] <- c(exp(fit.sim$par[1]), plogis(fit.sim$par[2]),
                       exp(fit.sim$par[3]))
    hess.sim <- optimHess(fit.sim$par, scr.nll, capt = capt.sim,
                          traps = test.data$traps, mask = test.data$mask)
    vcov.link.sim <- solve(hess.sim)
    ses.link.sim <- sqrt(diag(vcov.link.sim))
    ## For D:
    D.cis[i, ] <- exp(fit.sim$par[1] + c(-1, 1)*qnorm(0.975)*ses.link.sim[1])
    ## For g0:
    g0.cis[i, ] <- plogis(fit.sim$par[2] + c(-1, 1)*qnorm(0.975)*ses.link.sim[2])
    ## For sigma:
    sigma.cis[i, ] <- exp(fit.sim$par[3] + c(-1, 1)*qnorm(0.975)*ses.link.sim[3])
}
save(par.ests, D.cis, g0.cis, sigma.cis, file = "sim-res.RData")
```

(a) How close are the averages of your parameter estimates to the true parameter values?

(b) For STATS 730 graduates only: How often do your confidence intervals capture the true parameter values?

```r
## Averages across all simulations. Note that the model fit on the first
## iteration did not converge properly and gave me a silly answer, so I
## discarded it.
apply(par.ests[-1, ], 2, mean)

## [1]  0.7992249  0.8735727 73.5964857

## CI coverage for each parameter.
mean(D.cis[-1, 1] <= 0.75 & D.cis[-1, 2] >= 0.75)

## [1] 0.989899

mean(g0.cis[-1, 1] <= 0.9 & g0.cis[-1, 2] >= 0.9)

## [1] 0.9494949

mean(sigma.cis[-1, 1] <= 75 & sigma.cis[-1, 2] >= 75)

## [1] 0.8888889
```

The averages for all three parameters are close to the true parameter values. Our estimators appear more-or-less unbiased. With only 100 interations, it's hard to determine whether or not CI coverage reaches the nominal 95%, but all three are more or less there. We'd get a better indication with a larger number of iterations.

Questions for `ascr` users

9. Fit the same model from `secr-ll.r`, but using the `ascr` package. Verify that you get the same parameter estimates. For STATS 730 graduates, also verify that you get similar standard errors and confidence intervals.

```r
library(ascr)
## Fitting the model.
fit.ascr <- fit.ascr(capt = list(bincapt = test.data$bin.capt),
                     traps = test.data$traps, mask = test.data$mask)
## Looking at the estimates and standard errors.
summary(fit.ascr)

## Detection function: Halfnormal
```

```
## Information types:
##
##  Parameters:
##         Estimate Std. Error
## D       0.30895     0.2326
## g0      0.70979     0.1582
## sigma 157.48847    68.3650
## ---
##
## esa.1  45.31470    31.9020
```

```
## Comparison with our estimates and standard errors.
cbind(c(exp(fit$par[1]), plogis(fit$par[2]), exp(fit$par[3])),
      ses.unlink)
```

```
##                   ses.unlink
## [1,]   0.3087774  0.2327907
## [2,]   0.7097058  0.1582687
## [3,] 157.5819451 68.4868909
```

```
## Looking at the confidence intervals.
confint(fit.ascr, linked = TRUE)
```

```
##             2.5 %     97.5 %
## D      0.07061558   1.351693
## g0     0.35181792   0.916809
## sigma 67.25772513 368.769805
```

```
## Comparison with our confidence intervals.
rbind(exp(fit$par[1] + c(-1, 1)*qnorm(0.975)*ses.link[1]),
      plogis(fit$par[2] + c(-1, 1)*qnorm(0.975)*ses.link[2]),
      exp(fit$par[3] + c(-1, 1)*qnorm(0.975)*ses.link[3]))
```

```
##             [,1]        [,2]
## [1,]  0.07045558   1.3532426
## [2,]  0.35167081   0.9167979
## [3,] 67.23024051 369.3586286
```

Our estimates, standard errors, and confidence intervals are all very similar. Note that, by default, `ascr` appears to construct the confidence intervals directly on the untransformed parameters, which isn't entirely sensible! You need to add the argument `linked = TRUE` to calculate confidence intervals for the transformed parameters, and then convert them to the untransformed parameters. I should probably change the default behaviour.

Questions for `secr` users

10. Fit the same model from `secr-ll.r`, but using the `secr` package. Verify that you get the same parameter estimates. For STATS 730 graduates, also verify that you get similar standard errors and confidence intervals. This will require some data reformatting.

```
library(secr)
## Reformatting the data.
capt.secr <- convert.capt.to.secr(list(bincapt = test.data$bin.capt),
                                      traps = test.data$traps)
mask.secr <- convert.mask(test.data$mask)
traps.secr <- convert.traps(test.data$traps)
## Fitting the model.
fit.secr <- secr.fit(capthist = capt.secr, mask = mask.secr, trace = FALSE)
```

```r
## Looking at the estimates, standard errors, and confidence intervals.
predict(fit.secr)
```

```
##        link    estimate SE.estimate         lcl         ucl
## D       log   0.3089609   0.2702143  0.07052337   1.3535491
## g0    logit   0.7097908   0.1582217  0.35180743   0.9168151
## sigma   log 157.4852367  71.7571148 67.22581532 368.9296985
```

```r
## Comparison with our estimates and standard errors.
cbind(c(exp(fit$par[1]), plogis(fit$par[2]), exp(fit$par[3])),
      ses.unlink)
```

```
##                  ses.unlink
## [1,]   0.3087774  0.2327907
## [2,]   0.7097058  0.1582687
## [3,] 157.5819451 68.4868909
```

```r
## Comparison with our confidence intervals.
rbind(exp(fit$par[1] + c(-1, 1)*qnorm(0.975)*ses.link[1]),
      plogis(fit$par[2] + c(-1, 1)*qnorm(0.975)*ses.link[2]),
      exp(fit$par[3] + c(-1, 1)*qnorm(0.975)*ses.link[3]))
```

```
##             [,1]        [,2]
## [1,]  0.07045558   1.3532426
## [2,]  0.35167081   0.9167979
## [3,] 67.23024051 369.3586286
```

The standard errors are a little different for some reason, but the estimates and confidence intervals are very similar.

References

Efford, M. G., Dawson, D. K., & Borchers, D. L. (2009). Population density estimated from locations of individuals on a passive detector array. *Ecology*, *90*, 2676–2682.