Appendix 1: Implementation in the R software

The likelihood for an ordinary spatial capture-recapture model in the absence of class structure is specified by the R function `intlik2` from Royle et al. (2014):

```
intlik2 <- function(parm,y=y,X=traplocs,delta=.3,ssbuffer=2){

   Xl <- min(X[,1]) - ssbuffer
   Xu <- max(X[,1]) + ssbuffer
   Yu <- max(X[,2]) + ssbuffer
   Yl <- min(X[,2]) - ssbuffer

   xg <- seq(Xl+delta/2,Xu-delta/2,delta)
   yg <- seq(Yl+delta/2,Yu-delta/2,delta)
   npix.x <- length(xg)
   npix.y <- plength(yg)

   G <- cbind(rep(xg,npix.y),sort(rep(yg,npix.x)))
   nG <- nrow(G)
   D <- e2dist(X,G)
   area<- (delta*delta)*nrow(G)

   # extract the parameters from the input vector
   alpha0 <- parm[1]
   alpha1 <- exp(parm[2])
   n0 <- exp(parm[3])  # note parm[3] lives on the real line
   probcap <- plogis(alpha0)*exp(-alpha1*D*D)
   Pm <- matrix(NA,nrow=nrow(probcap),ncol=ncol(probcap))
   ymat <- rbind(y,rep(0,ncol(y))) ## tack on all-zero history
   lik.marg <- rep(NA,nrow(ymat))
   for(i in 1:nrow(ymat)){
     Pm[1:length(Pm)] <- dbinom(rep(ymat[i,],nG), K,
                   probcap[1:length(Pm)], log=TRUE))
     lik.cond <- exp(colSums(Pm))
     lik.marg[i] <- sum( lik.cond*(1/nG) )
   }
   nv <- c(rep(1,length(lik.marg)-1),n0)
## part1 here is the combinatorial term,
##  log(factorial(N)) = lgamma(N+1)
   part1 <- lgamma(nrow(y)+n0+1) - lgamma(n0+1)
   part2 <- sum(nv*log(lik.marg))
   return( -1*(part1+ part2) )
}
```

This function uses an encounter probability model that is proportional to the kernel of a Gaussian probability distribution function ("half-normal" in distance sampling),

$$p(\mathbf{x},\mathbf{s}) = \text{logit}^{-1}(\alpha_0)\exp(-\alpha_1 \| \mathbf{x}-\mathbf{s} \|^2)$$

where $\text{logit}^{-1}(\alpha_0)$ is the baseline encounter probability in trap $\mathbf{x}$, $\| \mathbf{x}-\mathbf{s} \|$ is the distance between trap $\mathbf{x}$ and activity center $\mathbf{s}$, and $\alpha_1 = 1/(2\sigma^2)$. In the optimization of the likelihood we estimate the parameters α_0 and $\log(\alpha_1)$ for numerical stability.

The function takes an argument, the unknown parameters of the model, and additional arguments including the encounter history matrix $\mathbf{y}$, the trap locations `traplocs`, the spacing of the integration grid (argument `delta`) and the state-space buffer. The first few lines of the function compute a grid of points used to evaluate the integrand, and the remaining half of the function computes the marginal likelihood contributions for each individual and the "all zero" encounter history. R code for simulating data and optimizing the likelihood using this function is in the `scrbook R` package from Royle et al. (2014) which can be viewed by looking at the help file `?intlik2`.

In the following R script, we provide some functions for simulating data for a 2-class situation (e.g., sex structured models) and a modification of the `intlik2` function to fit models with or without sex. The function `simSCR0sex` simulates sex-structured data along

with a given fraction of missing data (argument `na.frac`). The simulation function also simulates data with a behavioral response. The more general likelihood function is provided in the `R` package `scrbook` (Royle et al. 2014) available from the authors.

```
### This function simulates data with a sex-specific
### intercept and some missing sex values (na.frac), also
### allows for a behavioural response (beta).

e2dist<- function (x, y){
    i <- sort(rep(1:nrow(y), nrow(x)))
    dvec <- sqrt((x[, 1] - y[i, 1])^2 + (x[, 2] - y[i, 2])^2)
    matrix(dvec, nrow = nrow(x), ncol = nrow(y), byrow = F)
}

simSCR0sex <- function (N = 100, K = 20, alpha0 = c(-2.5,
  -1.8), sigma = c(0.5,0.75), beta=0, discard0 = TRUE,
  array3d = FALSE, psi.sex=.35,theta = .9){
### beta = behavioral response
### theta = PER SAMPLE probability of obtaining sex info
    traplocs <- cbind(sort(rep(1:7, 7)), rep(1:7, 7))
    Dmat <- e2dist(traplocs, traplocs)
    ntraps <- nrow(traplocs)
    plot(traplocs)
    buffer <- 2
    Xl <- min(traplocs[, 1] - buffer)
    Xu <- max(traplocs[, 1] + buffer)
    Yl <- min(traplocs[, 2] - buffer)
    Yu <- max(traplocs[, 2] + buffer)
    sx <- runif(N, Xl, Xu)
    sy <- runif(N, Yl, Yu)
    sex<- rbinom(N,1,psi.sex)
    if(length(alpha0) != 2){
         alpha0<- c(alpha0,alpha0)
    }
    S <- cbind(sx, sy)
    D <- e2dist(S, traplocs)
    if(length(sigma) != 2){
        sigma<- c(sigma, sigma)
    }

    alpha1 <- 1/(2 * sigma * sigma)
    probcap1 <- plogis(alpha0[1])*exp(-alpha1[1] * D * D)
    probcap2 <- plogis(alpha0[2])*exp(-alpha1[2] * D * D)
```

```
    Y <- matrix(NA, nrow = N, ncol = ntraps)
    for(i in 1:nrow(Y)){
     if(sex[i]==0)
       Y[i, ] <- rbinom(ntraps, K, probcap1[i, ])
     if(sex[i]==1)
       Y[i, ] <- rbinom(ntraps, K, probcap2[i, ])
    }
    if(discard0) {
      ncaps <- apply(Y, 1, sum)
      Y <- Y[ncaps > 0, ]
    }
    dimnames(Y)<- list(1:nrow(Y), paste("trap", 1:ncol(Y), sep=""))
    if(array3d){
      Y <- array(NA, dim = c(N, ntraps, K))
     for (i in 1:nrow(Y)) {
      for (j in 1:ntraps) {
        prevcap<- 0
        Y[i,j,1]<- rbinom(1,1,plogis(alpha0[sex[i]+1] )*exp(-
alpha1[sex[i]+1]*D[i,j]*D[i,j]) )
      for(k in 2:K){
        prevcap<- sum(Y[i,j,1:(k-1)])>0
        Y[i,j,k]<- rbinom(1,1,plogis(alpha0[sex[i]+1] + beta*prevcap)*exp(-
alpha1[sex[i]+1]*D[i,j]*D[i,j]) )
               }
           }
        }
        if (discard0) {
            Y2d <- apply(Y, c(1, 2), sum)
            ncaps <- apply(Y2d, 1, sum)
            Y <- Y[ncaps > 0, , ]
        }
    }
    sex.obs<-sex[ncaps>0]

    p.obtain.sex<- 1-(1-theta)^ncaps[ncaps>0]
    obtain.sex<- rbinom(length(sex.obs),1,p.obtain.sex)
    sex.obs[!obtain.sex]<- NA
    list(Y = Y, traplocs = traplocs, xlim = c(Xl, Xu), ylim = c(Yl,
        Yu), N = N, alpha0 = alpha0, alpha1 = alpha1, sigma = sigma,
        K = K,sex=sex.obs)
}


### Here is a likelihood function patterned off of intlik2
### from the "scrbook" package. Takes argument "sex" (observed data)
### parm should be 5 in length:
### alpha0[1] alpha0[2], log(alpha1), log(n0), logit(psi.sex)
```

```
intlik2sex<-
function (parm, y = y, delta = 0.3, K, X = traplocs, sex, ssbuffer =
2,sex.int=TRUE,sex.sig=TRUE)
{

    sex<-sex+1
    Xl <- min(X[, 1]) - ssbuffer
    Xu <- max(X[, 1]) + ssbuffer
    Yu <- max(X[, 2]) + ssbuffer
    Yl <- min(X[, 2]) - ssbuffer
    xg <- seq(Xl + delta/2, Xu - delta/2, delta)
    yg <- seq(Yl + delta/2, Yu - delta/2, delta)
    npix.x <- length(xg)
    npix.y <- length(yg)

    G <- cbind(rep(xg, npix.y), sort(rep(yg, npix.x)))
    nG <- nrow(G)
    D <- e2dist(X, G)

    area<- (delta*delta)*nrow(G)

    if(!sex.sig & sex.int){
     alpha0<- parm[1:2]
     alpha1 <- c(exp(parm[3]),exp(parm[3]))
     n0 <- exp(parm[4])
     psi.sex<-plogis(parm[5])
    }

    if(sex.sig & sex.int){
      alpha0<- parm[1:2]
      alpha1 <- exp(parm[3:4])
      n0 <- exp(parm[5])
      psi.sex<-plogis(parm[6])
    }
   if(sex.sig & !sex.int){
      alpha0<- c(parm[1],parm[1])
      alpha1<- exp(parm[2:3])
      n0 <- exp(parm[4])
      psi.sex<-plogis(parm[5])
   }
   if(!sex.sig & !sex.int){
     alpha0<- c(parm[1],parm[1])
     alpha1<- c(exp(parm[2]),exp(parm[2]))
     n0 <- exp(parm[3])
     psi.sex<-plogis(parm[4])
   }

    probcap1 <- plogis(alpha0[1]) * exp(-alpha1[1] * D * D)
    probcap2 <- plogis(alpha0[2]) * exp(-alpha1[2] * D * D)
```

```
    Pm1<-Pm2 <- matrix(NA, nrow = nrow(probcap1), ncol = ncol(probcap1))
    ymat <- y
    ymat <- rbind(y, rep(0, ncol(y)))
    lik.marg1<-lik.marg2 <- rep(NA, nrow(ymat))
    lik.marg<- lik.marg1
    for (i in 1:nrow(ymat)) {

        Pm1[1:length(Pm1)] <- (dbinom(rep(ymat[i, ], nG), rep(K,nG),
probcap1[1:length(Pm1)],
            log = TRUE))
        lik.cond1 <- exp(colSums(Pm1))
        lik.marg1[i] <- sum(lik.cond1 * (1/nG))

        Pm2[1:length(Pm2)] <- (dbinom(rep(ymat[i, ], nG), rep(K,nG),
probcap2[1:length(Pm2)],
            log = TRUE))
        lik.cond2 <- exp(colSums(Pm2))
        lik.marg2[i] <- sum(lik.cond2 * (1/nG))

        if(sex[i]==1 & !is.na(sex[i]))
        lik.marg[i]<- lik.marg1[i]*(1-psi.sex)

        if(sex[i]==2 & !is.na(sex[i]))
        lik.marg[i]<- lik.marg2[i]*psi.sex

        if(is.na(sex[i]))
        lik.marg[i]<- lik.marg1[i]*(1-psi.sex) + lik.marg2[i]*psi.sex

    }

    nv <- c(rep(1, length(lik.marg) - 1), n0)
    part1 <- lgamma(nrow(y) + n0 + 1) - lgamma(n0 + 1)
    part2 <- sum(nv * log(lik.marg))
    -1 * (part1 + part2)
}
```

The following set of `R` commands simulate a data set and fit the sex-structured model. The nature of the sex-specificity of model parameters is determined by the logical flags `sex.int` and `sex.sig`. If one or the other are `TRUE` then either or both the intercept or σ, respectively, are estimated as sex-specific parameters.

```
## Simulate a data set with no missing sex values and
## sex-specific intercepts but NOT sex-specific sigma
## Note "rnd" is the random number seed.
```

```
data<-simSCR0sex(N=200,discard0=TRUE,alpha0=c(-2.5,-1.8),
sigma=c(.75,.75), beta=0,theta=1, psi.sex=0.50)

## Harvest the elements of the data set
y<-data$Y
traplocs<-data$traplocs
nind<-nrow(y)
J<-nrow(traplocs)
K<-data$K
sex<-data$sex

##
##  First we obtain the MLEs for a model with sex-specific intercept
##   but not sex-specific sigma.

frog1<-nlm(intlik2sex,c(-2.5, -2, 1.0, log(4),0),
hessian=TRUE,y=y,X=traplocs, K=rep(K,J),
           sex=sex,delta=.3,ssbuffer=2,sex.int=TRUE,sex.sig=FALSE)

##
## Next we obtain MLEs for a model with NO sex-specificity but we
##  still estimate the probability of sex = 1

frog2<-nlm(intlik2sex,c(-2.5,1.5,log(4),0), hessian=TRUE,y=y,
X=traplocs, K=rep(K,J),
           sex=sex,delta=.3,ssbuffer=2, sex.int=FALSE,sex.sig=FALSE)
```