SEGUIMIENTO DE LA DIVERSIDAD BIOLÓGICA

Marcaje-reavistamiento espacialmente explícito (SMR)

José Jiménez García-Herrera (IREC-CSIC)

Universidad de Castilla-La Mancha

Los modelos de marcaje-reavistamiento espacialmente explícito (SMR) son extensiones de los SCR que se utilizan en los casos donde no todos los individuos se pueden diferenciar pero sí se diferencian individualmente una parte de ellos, y se puede hacer una separación clara entre una fracción marcada de la población (por marcas naturales o por marcas artificiales, como collares, crotales, etc) y la fracción no marcada (Sollmann et al., 2013). Atención que no se trata de usar individuos reconocidos por marcas y otros no reconocidos, sino **marcados** y **no marcados**. En caso de reconocidos y no reconocidos, lo que sería adecuado es usar la nueva extensión de SCR recientemente publicada: random thinning-SCR (Jiménez et al., 2020). Como en SMR solo se pueden emplear marcados y no marcados, hay una parte de las fotos que van a ser dudosas (no sabemos si los animales que aparecen están en una u otra de las dos categorías). Los eventos de asignación dudosa se desechan y no se usan en los análisis.

Repetid los análisis que se ejecutan aquí, cambiando cada uno de vosotros el número de individuos marcados, los valores del número de ocasiones de muestreo K y el valor de *rnd* que controla la aleatoriedad de la simulación. Para ampliar conocimientos resulta adecuado el libro *Spatial Capture-Recapture* de Royle et al. (2017) https://www.sciencedirect.com/book/9780124059399/spat capture-recapture.

1. Simulación de datos

Vamos a simular datos en `R` y a ejecutar el modelo SMR. En este caso vamos a simular **50 individuos** (que será la población a estimar), de los cuales hemos marcado previamente a **15 individuos** (que será la parte de la que vamos a tener información a nivel individual). Este modelo asume que que los animales marcados son un conjunto de individuos extraído aleatoriamente del espacio de estados, lo cual va a ser discutible si las capturas en vivo no están repartidas por todo el espacio de estados. Para superar esto, se suele utilizar el SMR-generalizado (Whittington, 2017), que incluye en el modelo el proceso de marcaje. No he mostrado este modelo por simplificar el aprendizaje, pero es recomendable conocerlo.

También es frecuente que haya animales para los cuales se pueda ver que van marcados, pero que éstas marcas no sean legibles. Para estos casos de lectura imperfecta, recomiendo la aproximación de Jiménez et al. (2019)

El nivel de precisión que podemos obtener con estas estimas va a depender del número de marcados, y del número de recapturas, que a su vez depende de la detectabilidad (λ_0), el número de ocasiones de muestreo K y la relación entre la distancia entre trampas/cámaras y el movimiento de los animales (σ). Para generar los datos simulados vamos a cargar varios paquetes de `R` y un conjunto de funciones que he compilado y modificado para ser utilizadas aquí.

```
> source("SCR_functions.R")  # Funciones de interés
> library(scrbook)           # Atención, scrbook no está en CRAN
> library(spatstat)
> library(lattice)
> library(coda)
```

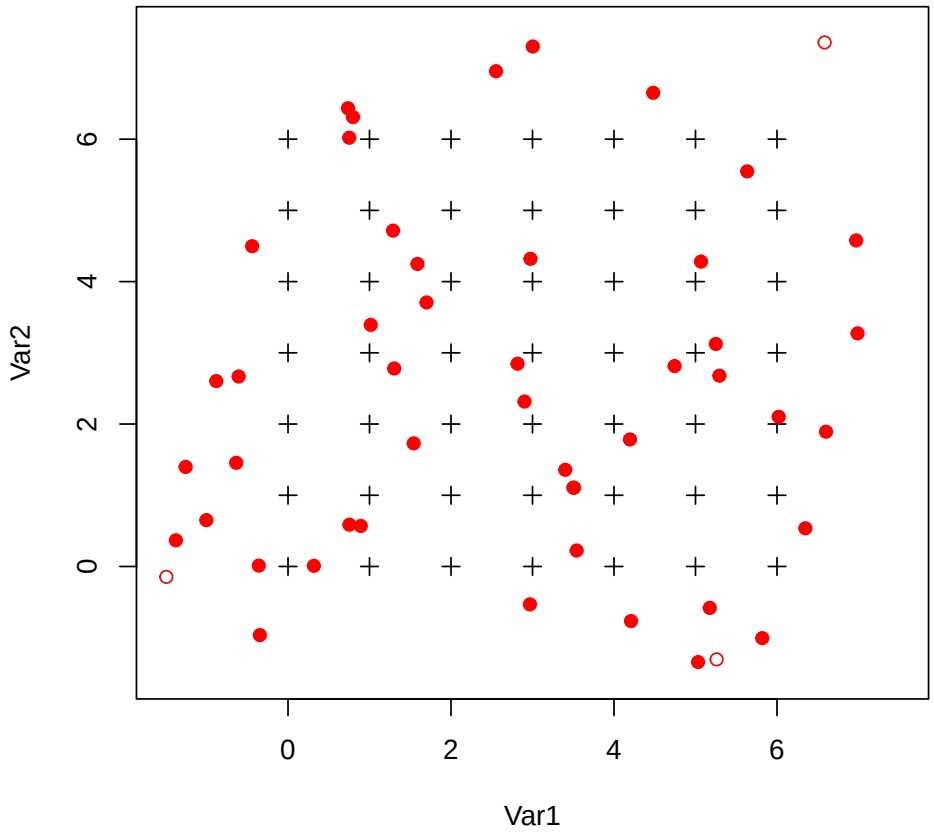
Datos a simular. Aquí debeis cambiar vosotros el valor de K.

```
> set.seed(1960)
> N <- 50  # Tamaño de población
> m <- 15  # marcados
> K <- 5   # Ocasiones de muestreo
> sigma <- 1.2 # parámetro de movimiento
> lam0 <- 0.2  # detectabilidad basal
```

Vamos a generar nuestros datos. Los veremos gráficamente y exploraremos con *str*:

```
> # Creamos trampas
> gx<-gy<-seq(0,6,1)
> X<-as.matrix(expand.grid(gx,gy))
> J<-dim(X)[1]
> # Límites de S
> xlims<-ylims<-c(-1.5, 7.5)
> # Simulador
> dat<-sim.pID.Data(N=N, K=K, sigma=sigma, lam0=lam0, knownID=m, X=X,
+   xlims=xlims, ylims=ylims, obsmod="pois", nmarked="known")
> plot(X, pch=3, xlim=xlims, ylim=ylims)
> points(dat$S, col="red", pch=1)
> S<-dat$S
```

```
> Y <- dat$Y; yt<-apply(Y, c(1,2), sum)
> detections <- rowSums(yt)>0
> points(S[detections,], pch=16, col="red")
> S<-dat$S
> Y <- dat$Y; yt<-apply(Y, c(1,2), sum)
> detections <- rowSums(yt)>0
> # Marcados
> yobs<- dat$Yobs
> # Quitamos las filas todos cero de la simulación (puede
> # haber marcados que no se detecten)
> yobs<-yobs[!apply(yobs,1,sum)==0,,]
> y <- apply(yobs,c(1,2),sum); sum(y)

[1] 80

> # Número de marcados
> (nMarked<-dim(y)[1])

[1] 14

> # Preparamos datos:
> # No marcados=Total-Marcados
> n<-dat$n-apply(dat$Yknown, 2:3, sum)
> n1<-apply(n,1,sum)
```

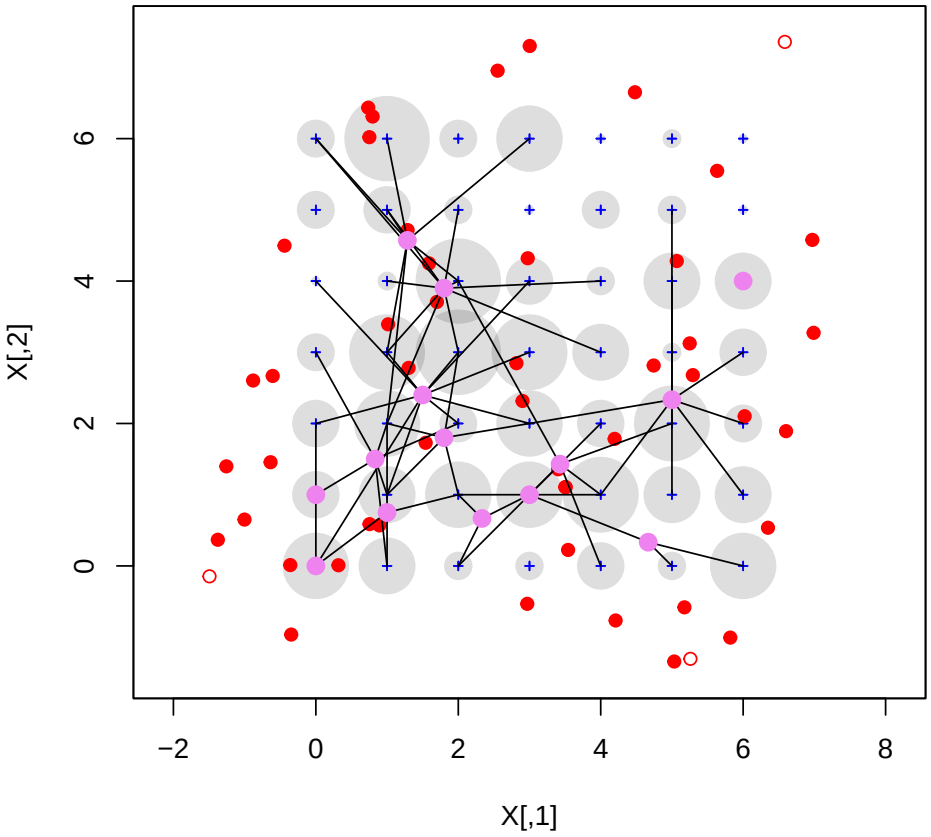


Os recuerdo que desechamos las fotos que no sabemos si corresponden a animales identificados o no. SMR no usa los registros de animales de estatus de identificación desconocido.

Ploteado de capturas

```
> X<-matrix(X, ncol=2)
> plot(X, pch=3, col="blue", cex=0.5, asp=TRUE,
+      xlim=xlims,
+      ylim=ylims, main="")
> tot<-apply(dat$n, 1,sum)
> symbols(X, circles=tot/15, inches=F,bg="#00000022", fg=NULL, add=T)
```

```
> points(X, pch=3, col="blue", cex=0.5)
> points(S, col="red", pch=1)
> points(S[detections,], pch=16, col="red")
> spiderplotJJ4(yobs, X, buffer=1.5, lwd=1)
```



2. Modelo en BUGS usando Nimble

```
> library(nimble)
> ## Definimos modelo
> code <- nimbleCode({
+
+   lam0 ~ dunif(0,5)
+   sig ~ dunif(0,5)
+   sig2 <- 2*sig^2
+   psi ~ dbeta(1,1)
+
+   for(i in 1:M) {
+     s[i,1] ~ dunif(xlim[1], xlim[2])
+     s[i,2] ~ dunif(ylim[1], ylim[2])
+     z[i] ~ dbern(psi)
+     d2[i,1:J] <- (s[i,1]-X[1:J,1])^2 + (s[i,2]-X[1:J,2])^2
+     lam[i,1:J] <- lam0*exp(-d2[i,1:J]/sig2)*z[i]*K
+   }
+
+   # Información de marcados, como en SCR
+   for(i in 1:nMarked) {
+     for(j in 1:J) {
+       y[i,j] ~ dpois(lam[i,j])
+     }
+   }
+
+   # Información de no marcados
+   for(j in 1:J) {
+     Lam[j] <- sum(lam[((nMarked+1):M),j])
+     n[j] ~ dpois(Lam[j])
+   }
+
+   N <- sum(z[1:M])
+   D <- N/A
+
+ })
```

Preparación de datos, constantes e inicios:

```
> # Inicios para las ubicaciones latentes
> M<-100
> A<-diff(xlims)*diff(ylims)
> sst<-cbind(runif(M, xlims[1], xlims[2]), runif(M, ylims[1], ylims[2]))
> for(i in 1:nMarked){
+   sst[i,1] <- mean( X[y[i,]>0,1] )
+   sst[i,2] <- mean( X[y[i,]>0,2] )
+ }
> str(constants <- list(M=M,              # aumentado de datos
+                       K=K,              # ocasiones de muestreo
+                       J=J,              # número de trampas
+                       nMarked=nMarked, # numero de marcados
+                       xlim=xlims,       # extremos x de S
+                       ylim=ylims,       # extremos y de S
+                       A=A))  # área del espacio de estados (S)

List of 7
 $ M      : num 100
 $ K      : num 5
 $ J      : int 49
 $ nMarked: int 14
 $ xlim   : num [1:2] -1.5 7.5
 $ ylim   : num [1:2] -1.5 7.5
 $ A      : num 81

> str( dataN  <-   list(y=y,   # matriz de capturas de marcados [i,j]
+                       X=X,   # coordenadas de las trampas [x,y]
+                       n=n1)) # vector de marcados por trampa [j]

List of 3
 $ y: int [1:14, 1:49] 0 0 0 1 0 1 0 0 0 1 ...
 $ X: num [1:49, 1:2] 0 1 2 3 4 5 6 0 1 2 ...
 $ n: num [1:49] 3 3 1 3 4 2 6 4 3 3 ...

> str( inits   <-  list(lam0=0.5, # probabilidad basal de detección
+                       sig=1,    # parametrización de sigma
+                       s=sst,    # ubicaciones de inicio
+                       z=c(rep(1, nMarked), rbinom((M-nMarked),1,0.2))))
```

```
List of 4
 $ lam0: num 0.5
 $ sig : num 1
 $ s   : num [1:100, 1:2] 3.43 3 1.8 1 1.29 ...
 $ z   : num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
```

3. Compilación y ejecución

```
> # Preparamos el modelo para ejecución en Nimble
> Rmodel <- nimbleModel(code=code,
+                       constants=constants,
+                       data=dataN,
+                       inits=inits,
+                       check=FALSE,
+                       calculate=FALSE)
> Cmodel <- compileNimble(Rmodel)
> # Establecemos los parámetros a monitorizar
> params <- c('psi', 'lam0', 'sig', 'N', 'D')
> mcmcspec<-configureMCMC(Rmodel, monitors=params)

===== Monitors =====
thin = 1: D, lam0, N, psi, sig
===== Samplers =====
RW sampler (202)
  - lam0
  - sig
  - s[]  (200 elements)
conjugate sampler (1)
  - psi
binary sampler (100)
  - z[]  (100 elements)

> # Cambiamos el muestreador de z (opcional)
> mcmcspec$removeSamplers('z')
> for(node in Rmodel$expandNodeNames('z')) mcmcspec$addSampler(target = node,
+                                                              type = 'slice')
> mcmcspec$removeSamplers("s")
> ACnodes <- paste0("s[", 1:constants$M, ", 1:2]")
```

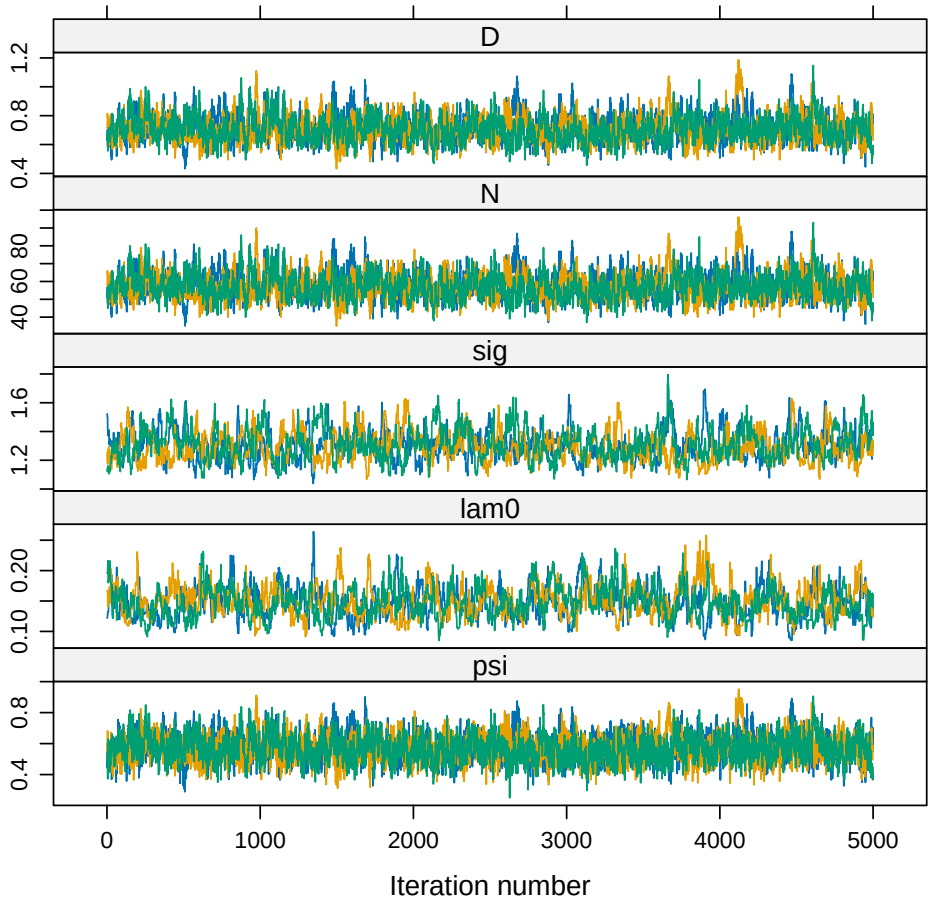
```
> for(node in ACnodes) {
+     mcmcspec$addSampler(target = node,
+                         type = "RW_block",
+                         control = list(adaptScaleOnly = TRUE),
+                         silent = TRUE)
+ }
> # Construimos el modelo
> scrMCMC <- buildMCMC(mcmcspec)
> # Compilamos
> CSCRMCMC <- compileNimble(scrMCMC, project = Rmodel)
> # Ejecutamos el modelo
> nb=1000      # Iteraciones a desechar
> ni=5000 +nb  # Iteraciones
> nc=3         # Cadenas
> start.time2<-Sys.time()
> outNim <- runMCMC(CSCRMCMC,
+                   niter = ni,
+                   nburnin = nb,
+                   nchains = nc,
+                   inits=inits,
+                   setSeed = TRUE,
+                   progressBar = TRUE,
+                   samplesAsCodaMCMC = TRUE)

|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|

> end.time<-Sys.time()
> end.time-start.time2 # tiempo de ejecución

Time difference of 3.782287 mins
```

4. Resultados

```
> summary(outNim[,c('psi', 'lam0', 'sig', 'N', 'D')])

Iterations = 1:5000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 5000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean      SD  Naive SE Time-series SE
psi   0.5736 0.08676 0.0007084       0.002713
lam0  0.1476 0.02490 0.0002033       0.001231
sig   1.3041 0.10258 0.0008376       0.004991
N    57.5315 7.41757 0.0605642       0.279189
D     0.7103 0.09157 0.0007477       0.003447

2. Quantiles for each variable:

        2.5%     25%     50%     75%   97.5%
psi   0.4138  0.5143  0.5698  0.6297  0.7505
lam0  0.1039  0.1306  0.1459  0.1626  0.2033
sig   1.1303  1.2307  1.2951  1.3684  1.5318
N    45.0000 52.0000 57.0000 62.0000 74.0000
D     0.5556  0.6420  0.7037  0.7654  0.9136
```

Inspeccionamos la convergencia de las cadenas de Markov

```
> xyplot(outNim[,c('psi', 'lam0', 'sig', 'N', 'D')])
```



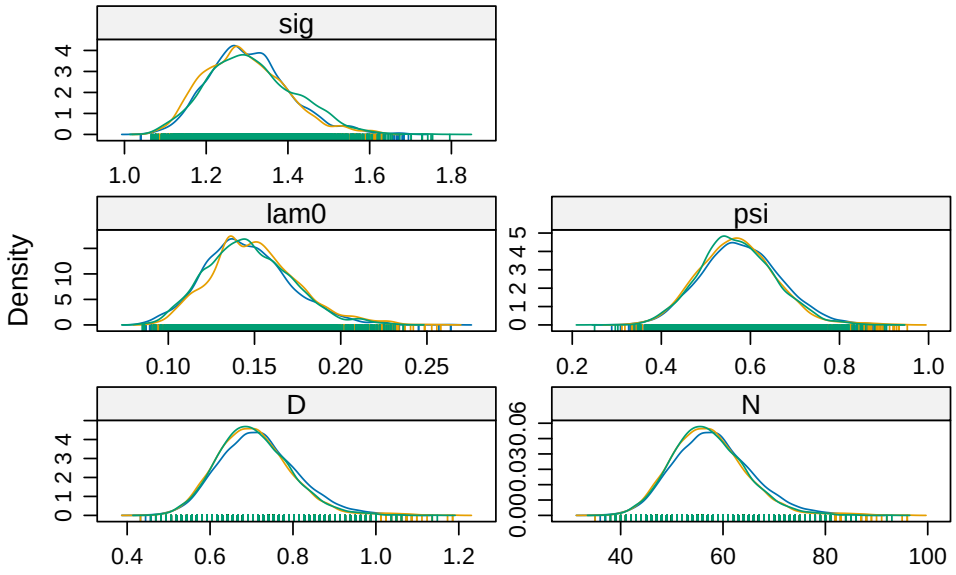

```
> gelman.diag(outNim, multivariate = FALSE)

Potential scale reduction factors:

     Point est. Upper C.I.
D          1.01       1.03
N          1.01       1.03
lam0       1.02       1.05
psi        1.01       1.03
sig        1.02       1.07
```

```
> samplesn<-data.matrix(outNim)
> # Coeficiente variación para N
> sd(samplesn[,2])/mean(samplesn[,2])
```

```
[1] 0.1289304
```

Para observar las distribuciones creadas

```
> densityplot(outNim)
```



5. REFERENCIAS

Jiménez, J., Augustine, B. C., Linden, D. W., Chandler, R. B., & Royle, J. A. (2020). Spatial capture–recapture with random thinning for unidentified encounters. *Ecology and Evolution*, ece3.7091. doi:10.1002/ece3.7091

Jiménez, J., Chandler, R. B., Tobajas, J., Descalzo, E., Mateo, R., & Ferreras, P. (2019). Generalized spatial mark–resight models with incomplete identification: An application to red fox density estimates. Ecology and Evolution, 9(8), ece3.5077. doi:10.1002/ece3.5077

R Core Team. (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from https://www.r-project.org/.

Royle, J. A., Chandler, R. B., Sollmann, R., & Gardner, B. (2014). *Spatial capture-recapture*. Waltham, Massachusetts: Elsevier, Academic Press. doi:10.1016/B978-0-12-405939-9.00026-8

Sollmann, R., Gardner, B., Parsons, A. W., Stocking, J. J., Mcclintock, B. T., Simons, T. R., ... O'Connell, A. F. (2013). A spatial mark-resight model augmented with telemetry data. *Ecology*, 94(3), 553–559. doi:10.1890/12-1256.1

Whittington, J., Hebblewhite, M., & Chandler, R. B. (2017). Generalized spatial mark-resight models with an application to grizzly bears. *Journal of Applied Ecology*, (November 2016), 1–12. doi:10.1111/1365-2664.12954