

SEGUIMIENTO DE LA DIVERSIDAD BIOLÓGICA

SCR-sin marcar o conteos Espaciales (UN-SCR)

José Jiménez García-Herrera (IREC-CSIC)

Universidad de Castilla-La Mancha

El último método que vamos a ver es el unmarked-SCR o conteos espaciales (UN-SCR). Los modelos de conteos espaciales son otra extensión de los SCR, para el caso de que ningún animal está marcado y por tanto es reconocible (Chandler y Royle, 2013). Se basa en la información suministrada por la correlación espacial entre capturas. Esta correlación tiene un valor informativo muy limitado, y esa limitación exige suministrar datos adicionales de telemetría, o priors informativo o bien la identificación de algún individuo. **Si no se usa con otra información adicional es un método de muy baja precisión.** Una cuestión importante a tener en cuenta es que sólo funciona realmente bien para bajas densidades.

Repetid los análisis que se ejecutan aquí, cambiando cada uno de vosotros 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/capture-recapture>.

1. Simulación de datos

Vamos a simular datos en R y a ejecutar el modelo UN-SCR. En este caso vamos a simular **20 individuos** (que será la población a estimar). Como os he comentado antes, sin información adicional, esta modelo es muy poco preciso. Para solventarlo, voy a utilizar aquí un prior informativo con una distribución gamma.

Para generar los datos simulados vamos a cargar varios paquetes de R y una simulación de datos que vamos a hacer directamente (no usaremos una función predefinida).

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

Datos a simular.

```
> tr<-seq(1.5,8.5, length=10)
> # 100 coord. trampas
> X<-cbind(rep(tr,each=length(tr)),rep(tr,times=length(tr)))
> set.seed(1965)
> xlim <- c(0,10); ylim <- c(0,10)
> N <- 20
> S <- cbind(runif(N, xlim[1], xlim[2]), runif(N, ylim[1], ylim[2]))
```

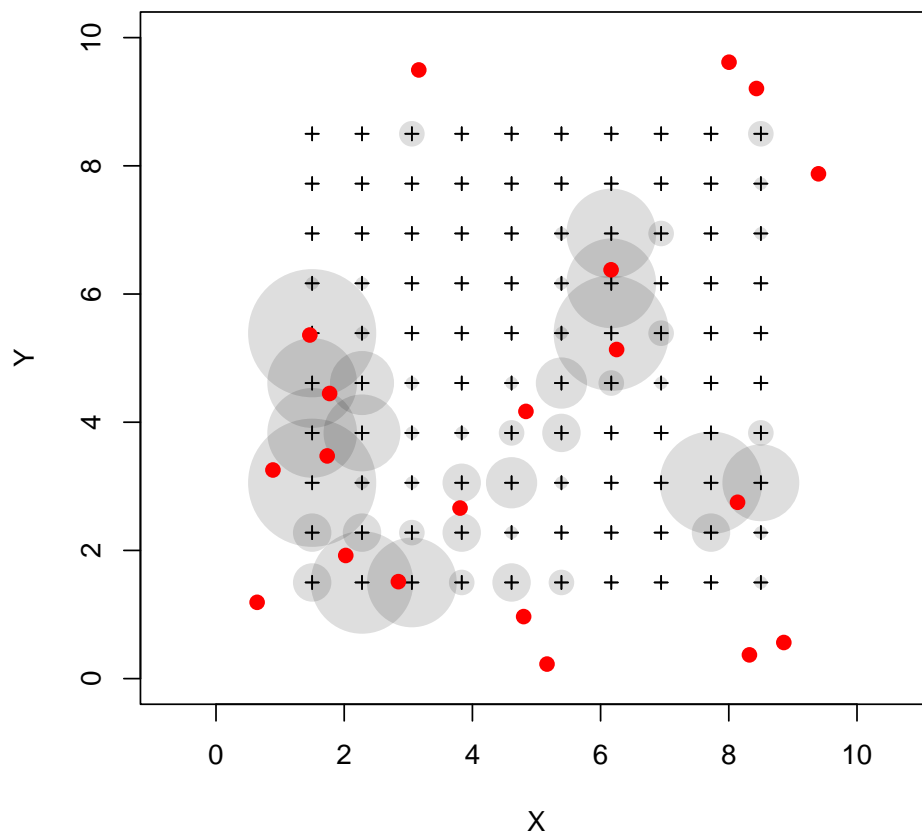
Vamos a generar nuestros datos:

```
> J <- nrow(X)
> K <- 15
> sigma <- 0.5
> lambda0 <- 0.4
> r<-3
> yy <- array(NA, c(N, J, K))
> for(j in 1:J) {
+   dist <- sqrt((X[j,1]-S[,1])^2 + (X[j,2]-S[,2])^2)
+   lambda <- lambda0*exp(-dist^2/(2*sigma^2))
+   for(k in 1:K) {
+     yy[,j,k] <- rpois(N, lambda)
+   }
+ }
> sum(yy)
[1] 168

> n <- apply(yy, c(2,3), sum);
> n[is.na(n)] <- 0
> n1<-apply(n, 1, sum)
> sum(n)
[1] 168

> M<-150
> plot(X, xlim=c(0,10), ylim=c(0,10), pch=3, cex=0.75,
+       xlab="X", ylab="Y", asp=TRUE)
> # Lo anadimos al ploteado:
> tot<-apply(n, 1,sum)
```

```
> symbols(X, circles=tot/10, inches=F, bg="#00000022",
+         fg=NULL, add=T)
> points(X, pch=3, cex=0.75); points(S, pch=16, col=2)
> points(S, pch=16, col="red", cex=1.25)
```



Vamos a usar un prior informativo para sigma, generandolo con una distribucion gamma. Si queremos usar un sigma:

```
> mode = 0.5
> sd = 0.1
> # Obtenemos los parámetros de ratio y forma:
```

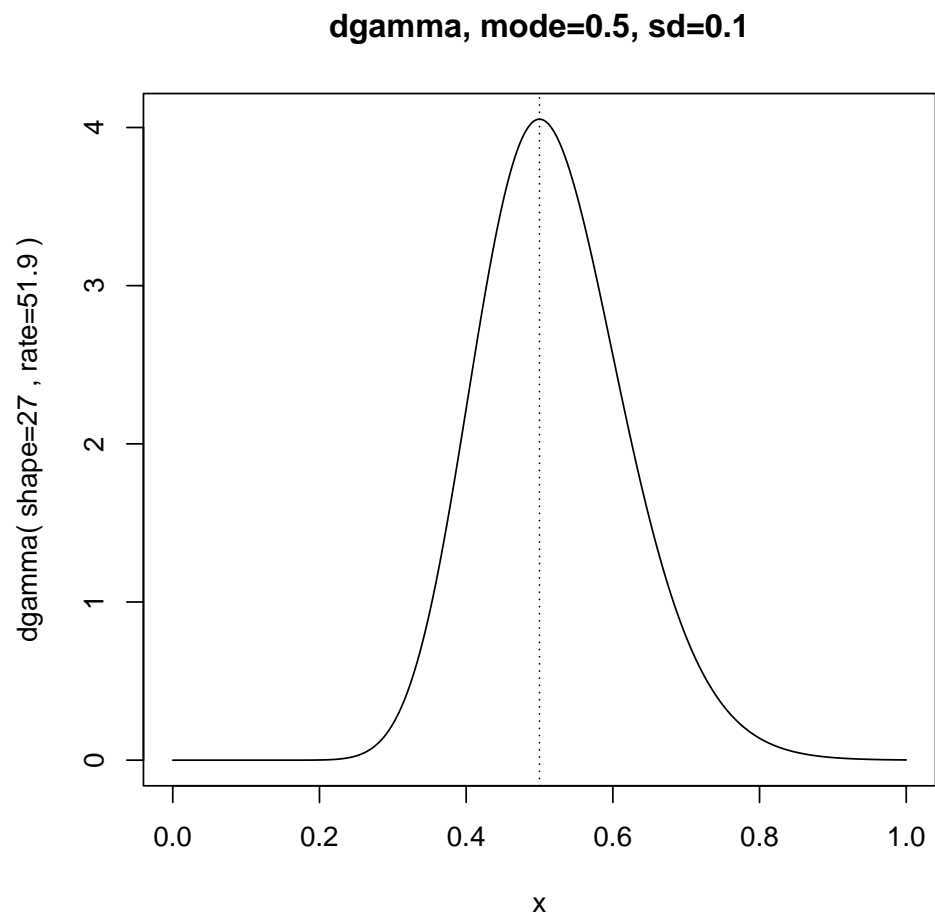
```
> ra = ( mode + sqrt( mode^2 + 4*sd^2 ) ) / ( 2 * sd^2 )
> sh = 1 + mode * ra
> show(sh)

[1] 26.96291

> show(ra)

[1] 51.92582

> # Ploteamos:
> x = seq(0,mode+5*sd,len=1001)
> plot( x , dgamma( x , shape=sh , rate=ra ) , type="l" ,
+       main=paste("dgamma, mode=",mode," , sd=",sd,sep="") ,
+       ylab=paste("dgamma( shape=",signif(sh,3)," , rate=",signif(ra,3)," )",
+                 sep="") )
> abline( v=mode , lty="dotted")
```



Código

```
> library(nimble)
> ## define the model
> code <- nimbleCode({
+
+   sigma ~ dgamma(sh,ra)
+   psi ~ dunif(0,1)
+   lam0 ~ dunif(0,5)
+
+   for(i in 1:M) {
+     z[i] ~ dbern(psi)
+     s[i,1] ~ dunif(xlim[1],xlim[2])
+     s[i,2] ~ dunif(ylim[1],ylim[2])
+     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]/(2*sigma^2))*z[i]*K
+   }
+
+   for(j in 1:J){
+     bigLambda[j] <- sum(lam[1:M,j])
+     n[j] ~ dpois(bigLambda[j])
+   }
+   N <- sum(z[1:M])
+
+ })
```

Constantes, datos e inicios

```
> constants <- list(M = M, K=K, J=J, sh=sh, ra=ra)
> data<-list(n=n1, X=X, xlim=xlim, ylim=ylim)
> s.start <- cbind(runif(M, 0, 10), runif(M, 0, 10))
> # Funciones de inicio
> d <- e2dist(s.start[1:M,], X)
> lam <- 0.2 * exp( -(d^2)/(2 * 0.5^2))
> yi <- array(0, c(M, J, K)) # resighting array
> for (j in 1:J) {
+   for (k in 1:K) {
```

```
+   if (n[j, k] > 0) {
+     probs <- lam[ ,j]
+     probs <- probs / sum(probs)
+     latent.id <- sample(1:M, n[j,k], prob = probs, replace = FALSE)
+     yi[latent.id , j, k] <- 1
+   }
+ } # end of k
+ } # end of j
> yis <- apply(yi, c(1,2), sum)
> ytot<- apply(yis,c(1,2),sum)
> ssarr<- array(NA,dim=c(M,2))
> for(i in 1:M){
+   if(sum(ytot[i,])==0) next
+   s.start[i,1]<- mean( X[ytot[i,]>0,1] )
+   s.start[i,2]<- mean( X[ytot[i,]>0,2] )
+ }
> ssarr[,]<- s.start
> z<-rep(1,M)
> inits <- list (sigma=5, lam0=0.1, lam=yis, s=s.start, z=z)
```

Ejecutamos el código:

```
> Rmodel <- nimbleModel(code=code, constants=constants,
+                       data=data, inits=inits,
+                       check=FALSE, calculate=FALSE)
> Cmodel <- compileNimble(Rmodel)
> params<-c('N', 'sigma', 'lam0', 'psi')
> mcmcspec<-configureMCMC(Rmodel,
+                          monitors=params,
+                          thin=1)

===== Monitors =====
thin = 1: lam0, N, psi, sigma
===== Samplers =====
RW sampler (303)
- sigma
- psi
```

```
- lam0
- s[] (300 elements)
binary sampler (150)
- z[] (150 elements)

> 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)
+ }
> pumpMCMC <- buildMCMC(mcmcspec)
> CpumpMCMC <- compileNimble(pumpMCMC, project = Rmodel)
> # Ejecutamos el modelo
> nb=1000      # Iteraciones a desechar
> ni=5000 +nb  # Iteraciones
> nc=3         # Cadenas
> outNim <- runMCMC(CpumpMCMC, niter = ni , nburnin = nb ,
+                  nchains = nc, inits=inits,
+                  setSeed = FALSE, progressBar = TRUE,
+                  samplesAsCodaMCMC = TRUE)

|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
```


Resultado

```
> summary(outNim)
```

```
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
N	22.0238	4.53756	0.0370490	0.216242
lam0	0.4087	0.07475	0.0006103	0.004145
psi	0.1515	0.04172	0.0003407	0.001539
sigma	0.4866	0.03687	0.0003010	0.002092

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
N	15.00000	19.0000	21.0000	25.0000	32.0000
lam0	0.26465	0.3601	0.4079	0.4561	0.5622
psi	0.08343	0.1218	0.1478	0.1765	0.2449
sigma	0.42319	0.4612	0.4830	0.5082	0.5690

```
> xyplot(outNim)
```

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

Potential scale reduction factors:

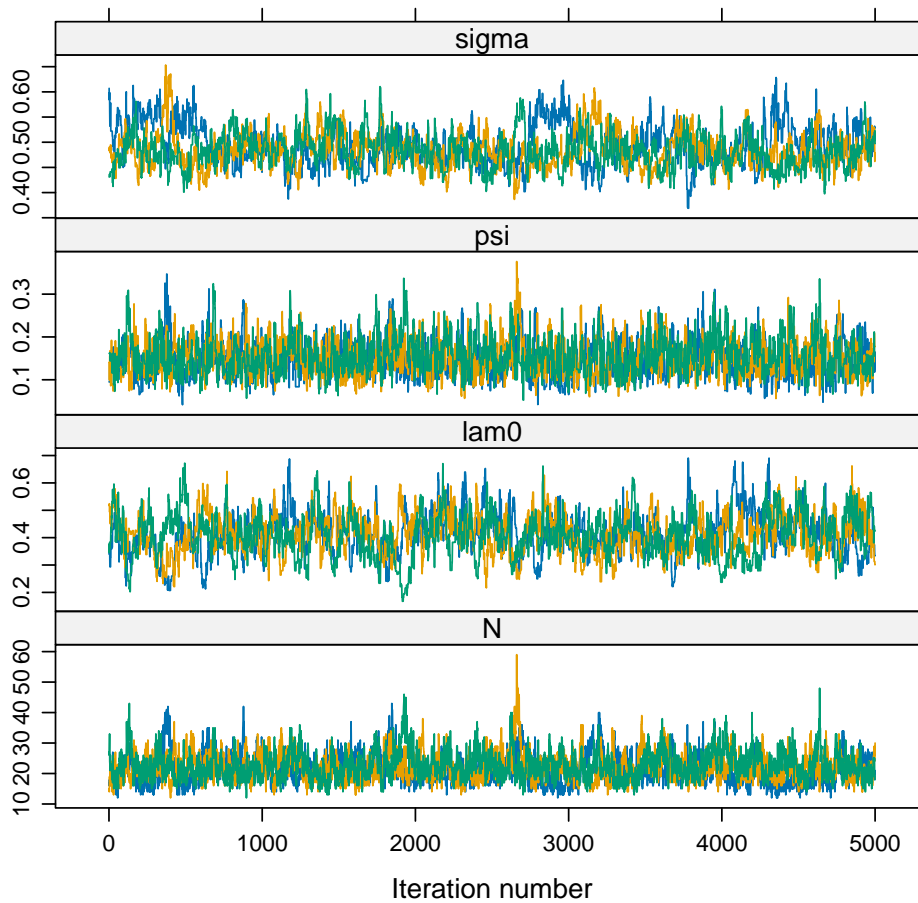
	Point est.	Upper C.I.
N	1.03	1.11
lam0	1.01	1.03
psi	1.02	1.06
sigma	1.06	1.16

```
> cat("Poblacion que simulamos = ", N, "individuos", "\n")
```

```
Poblacion que simulamos = 20 individuos
```

```
> cat("Fotografias (todos no identificados)", sum(n), "\n")
```

Fotografias (todos no identificados) 168



2. REFERENCIAS

- Chandler, R. B., & Royle, J. A. (2013). Spatially-explicit models for inference about density in unmarked populations. *The Annals of Applied Statistics*, 7(2), 936–954. doi:10.1214/12-AOAS610
- 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