Universidad de Castilla-La Mancha

# SEGUIMIENTO DE LA DIVERSIDAD BIOLÓGICA

## SCR con leopardo de las nieves

José Jiménez García-Herrera (IREC-CSIC)

*Universidad Pablo de Olavide*

Este es un análisis con datos reales. Utilizaréis archivos como los que vais a encontrar habitualmente, que se importan en formato `txt` o `csv`. Repetid primero los análisis que se ejecutan aquí para aseguraros de entender todo. Enviadme vuestras dudas por e-mail a: Jose.Jimenez@csic.es.

Librerías a utilizar

```
> library(scrbook)
> library(calibrate)
> library(aspace)
> library(nimble)
> library(coda)
> library(lattice)
> source("SCR_functions.R")
```

# Origen de los datos

Los datos que vamos a utilizar están extraídos de la página de Shannon Kachel (2016). Vamos a trabajar con SCR en `R` usando fundamentalmente la librería `scrbook` (atención, esta librería no está en CRAN, y hay que instalarla en local).

# Preparación y exploración de los datos

Vamos a cargar la información de las cámaras de fototrampeo, y vamos a representarla. Vamos a etiquetar las cámaras, y vamos a ver la distancia entre ellas. Para cargar la información que está en texto plano (`txt`) usamos el comando `read.table()`. Poned atención en que el path esté correctamente definido con `setwd()`.

## Cargamos la ubicación de las cámaras

```
> X1<-read.table("traps.txt", header=TRUE)
> dim(X1)
```

```
[1] 37  3
```

```
> # Veamos el contenido
> head(X1)
```

```
  ID      X       Y
1  1 427435 4170605
2  2 424254 4172691
3  3 431541 4178952
4  4 435207 4177925
5  5 437841 4174595
6  6 439218 4171834
```

```
> # Vamos a ver la ubicación de las trampas:
> xlims2<-c(min(X1[,2]),max(X1[,2]))+c(-1000,1000)
> ylims2<-c(min(X1[,3]),max(X1[,3]))+c(-1000,1000)
```

Y ahora vamos a estudiar las distancias entre las cámaras más próximas

```
> leopardo<-  X1[,2:3]
> output2 <- matrix(ncol=nrow(leopardo), nrow=nrow(leopardo))
> for (i in 1: nrow(leopardo)){
+   singlePoint <- c(leopardo[i,1], leopardo[i,2])
+   output2[i,] <- c(distances(centre.xy=singlePoint,
+                              destmat=leopardo, verbose=FALSE))
+   }
```

Convertimos en NA la diagonal

```
> output2[output2=="0"]<-NA
> DM<-apply(output2, 2, min, na.rm=TRUE); length(DM)
```

```
[1] 37
```

Distancia media entre trampas

```
> mean(DM, na.omit=TRUE)
```

```
[1] 2852.071
```

Rango de distancias

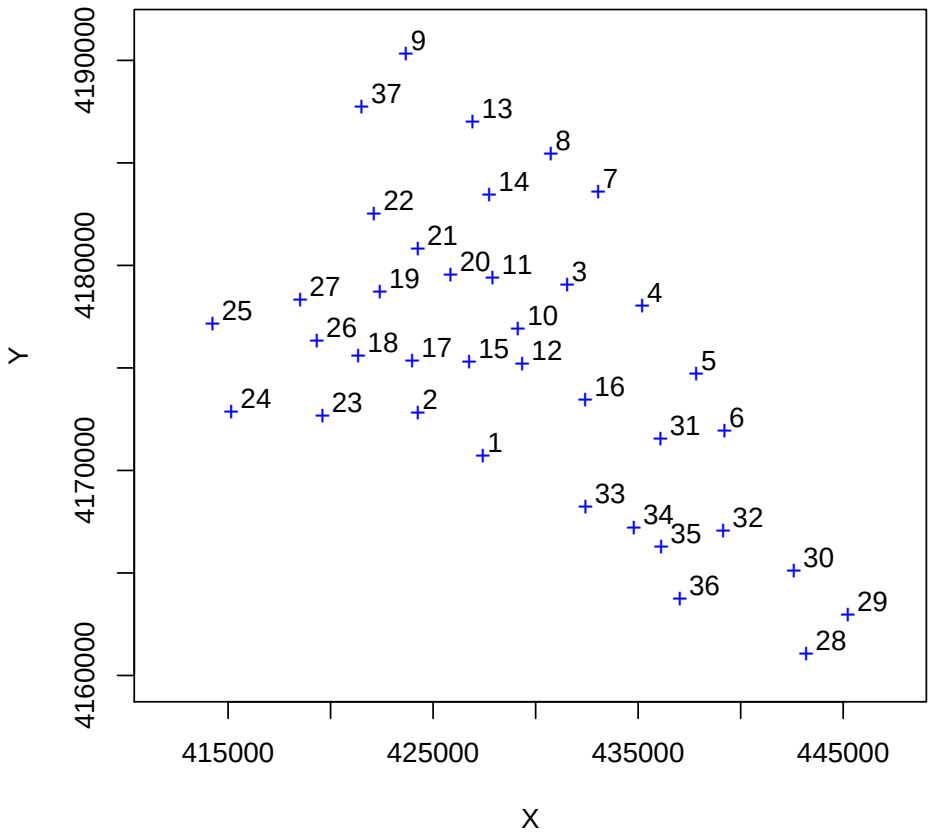
```
> range(DM)
```

```
[1] 1651.880 4357.926
```

¿Es adecuada la distancia entre trampas?... Lo veremos más tarde

Lo ploteamos usando asp=1 para que ajuste la relación x/y

```
> plot(X1[,2:3],pch="+",col="blue", asp=1, xlim=xlims2, ylim=ylims2)
> # Vamos a etiquetar las cámaras trampas:
> textxy(X1[,2], X1[,3], seq(1,37), m=c(0.5,0.5), cex=1)
```



Escalamos y centramos las coordenadas, ya que en general los programas que vamos a utilizar (WinBUGS, JAGS, Nimble,...) trabajan mejor alrededor del cero:

```
> X1 <- X1[,2:3]/1000
> names(X1)<-c('X','Y')
> # Estandarizamos
> newtrapsx<-X1[,1]-mean(X1[,1])
> newtrapsy<-X1[,2]-mean(X1[,2])
> X<-cbind(newtrapsx,newtrapsy)
> n.llx=min(X[,1])
> n.upx=max(X[,1])
> n.lly=min(X[,2])
> n.upy=max(X[,2])
> head(X)
```

```
     newtrapsx  newtrapsy
[1,] -1.707081 -4.6463243
[2,] -4.888081 -2.5603243
[3,]  2.398919  3.7006757
[4,]  6.064919  2.6736757
[5,]  8.698919 -0.6563243
[6,] 10.075919 -3.4173243
```

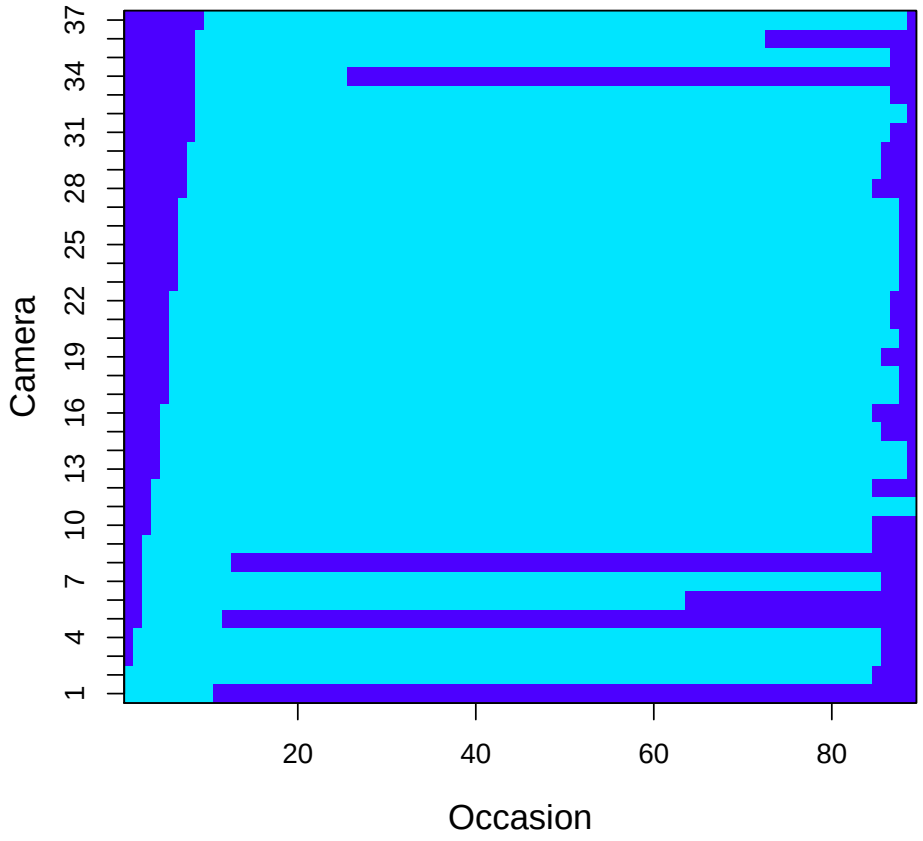
## Operatividad de las cámaras

Veamos la operatividad de las cámaras. Esta información la utilizaremos en el código SCR.

```
> Oper<-data.matrix(read.table("Oper.txt", header=FALSE))
> J<-dim(Oper)[1] # trampas
> K<-dim(Oper)[2] # dias de muestreo
```

Vamos a verlo gráficamente

```
> ## Tenemos que convertir esto en matriz
> x<-as.matrix(Oper)
> image(1:K,1:J,t(x), yaxt = "n", xlab="Occasion", ylab="",
+       col=topo.colors(2), cex.lab=1.25)
> mtext(side = 2, "Camera", line = 2.5, cex=1.25)
> axis(2, rev(seq(1, 43, by=1)))
> box()
```




Y apilamos los dias de operatividad de cada cámara

```
> KT<-apply(Oper,1,sum)
> colnames(Oper)<-c(1:K)
```

## Información y exploración de las capturas

Vamos a leer las capturas para ponerlas en un formato utilizable, y organizarlas para su uso en el codigo SCR:

```
> wtraps<-cbind(1:J,X,Oper)
> wcaps<-as.matrix(read.table("ID.txt", header=TRUE))
```

```
> head(wcaps)
```

```
     year individual day trap
[1,]    1          1   6    1
[2,]    1          1   3    4
[3,]    1          2   7    6
[4,]    1          2  27    6
[5,]    1          2  46    6
[6,]    1          3  52    7
```

Vamos a ver antes si hay algún duplicado (si trabajamos con una binomial, no puede haber dos capturas del mismo animal en la misma ocasión. Si las hubiera, quitamos una de las repeticiones. Esta operación se suele llamar *"thinning"*)

```
> wcaps[duplicated(wcaps),]
```

```
     year individual day trap
```

Ahora usamos una función que a partir de las trampas y las capturas crea una matriz tridimensional (individuo-trampa-ocasión)

```
> datYknown <-SCR23darray(wcaps, wtraps)
> str(datYknown)
```

```
 num [1:19, 1:37, 1:89] 0 0 0 0 0 0 0 0 0 0 ...
```

¿Cuantos individuos se han visto?

```
> nind<-dim(datYknown)[1]; nind # es la primera dimensión de la matriz 3D
```

```
[1] 19
```

Un error frecuente en los datos es que no coincidan la operatividad y las fechas de las observaciones (sería equivalente a que un dia que una cámara no está operativa se produce una captura). Esto, que tiene una repercusión muy grave en las estimas, se comprueba multiplicando la matriz de capturas por la operatividad. Primero pasamos la matriz 3d `[i,j,k]` a 2d `[j,k]`:

```
> y2d<-apply(datYknown, c(2,3),sum)
> sum(y2d)
```

```
[1] 70
```

```
> sum(y2d*Oper)
```

```
[1] 70
```

```
> # En este caso, al dar igual resultado, podemos pensar que no hay
> # discrepancias entre operatividad y capturas. Si hubiera un resultado
> # diferente, localizamos el error con esta instrucción:
> which((y2d*Oper)-y2d<0, arr.in=TRUE)
```

```
     row col
```

## Aumentado de datos

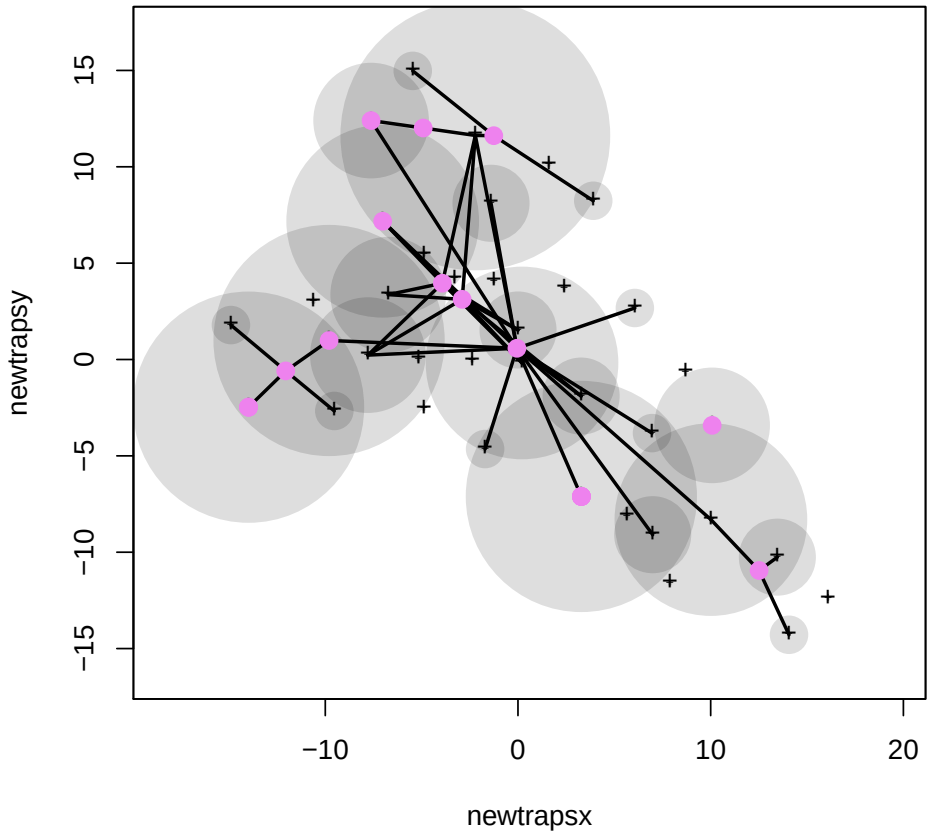
Creamos individuos "virtuales" que luego la ejecución del código nos va decir cuáles existen y cuáles no. En general se tiende a poner el triple de lo que uno piensa que es razonable que haya. Si el aumentado de datos fuera insuficiente, al crear el histograma del tamaño poblacional, lo percibiremos claramente truncado por la derecha. Si ponemos un aumentado demasiado alto, le ejecución va a ser muy lenta. En general, es mejor pasarse poniendo un número elevado en una ejecución de prueba, y luego ajustar en las siguientes.

```
> M<-100   # Ponemos más del triple de lo que a priori
>          # pensamos que podría haber.
> nind<- dim(datYknown)[1]
> Yaug <- array(0, dim = c(M, J, K))
> Yaug[1:nind, , ] <- datYknown
> y<-apply(Yaug, c(1,2), sum)
```

## Spiderplot

Vamos a plotear las capturas y la ubicación promedio de los diferentes individuos y los extremos donde se ha localizado (spiderplot)

```
> xlim<-c(n.llx,n.upx)
> ylim<-c(n.lly,n.upy)
> plot(X, xlim=xlim+c(-2,2), ylim=ylim+c(-2,2), pch="+", asp=TRUE)
> datn<-apply(datYknown, c(2,3), sum)
> tot<-apply(datn, 1,sum)
> symbols(X, circles=tot, inches=F, bg="#00000022", fg=NULL, add=T)
> points(X, pch="+", cex=0.5)
> # Spiderplot
> spiderplotJJ4(datYknown, X, buffer=2, lwd=2)
```




## Definición del espacio de estados

```
> buff<-20
> xlim<-xlim+c(-buff,buff)
> ylim<-ylim+c(-buff,buff)
> area<-diff(xlim)*diff(ylim)
```

# Código: definimos el modelo en BUGS

```
> ## Escribimos el modelo
> code <- nimbleCode({
+
+   alpha0 ~ dnorm(0,.01)
+   logit(p0) <- alpha0
+   alpha1 ~ dnorm(0,.01)
+   sigma <- sqrt(1/(2*alpha1))
+   psi ~ dunif(0,1)
+
+   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])
+     d[i,1:J] <- pow(s[i,1]-X[1:J,1],2) + pow(s[i,2]-X[1:J,2],2)
+     p[i,1:J] <- p0*z[i]*exp(- alpha1*d[i,1:J])
+
+     for(j in 1:J){
+       y[i,j] ~ dbin(p[i,j],KT[j])
+     }
+   }
+   N <- sum(z[1:M])
+   D <- 100*N/area    # individuos/100 km2
+ })
```

## Constantes

```
> constants <- list(M = M, K=K, J=J, area=area)
> str(constants)
```

```
List of 4
 $ M   : num 100
 $ K   : int 89
 $ J   : int 37
 $ area: num 4917
```

## Datos

```
> data <- list (y=y, X=X, xlim=xlim, ylim=ylim, KT=KT)
> str(data)
```

```
List of 5
 $ y   : num [1:100, 1:37] 1 0 0 0 0 0 0 0 0 0 ...
 $ X   : num [1:37, 1:2] -1.71 -4.89 2.4 6.06 8.7 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "newtrapsx" "newtrapsy"
 $ xlim: num [1:2] -34.9 36.1
 $ ylim: num [1:2] -34.3 35
 $ KT  : int [1:37] 10 84 84 84 9 61 83 10 82 81 ...
```

## Inicios

```
> z <- c(rep(1,nind),rep(0,M-nind))
> sst <- cbind(runif(M,xlim[1],xlim[2]),runif(M,ylim[1],ylim[2]))
> for(i in 1:nind){
+   sst[i,1] <- mean( X[y[i,]>0,1] )
+   sst[i,2] <- mean( X[y[i,]>0,2] )
+ }
> inits <- list (p0=0.01, alpha1=1, s=sst, z=z)
> str(inits)
```

```
List of 4
 $ p0    : num 0.01
 $ alpha1: num 1
 $ s     : num [1:100, 1:2] -0.054 10.076 -1.256 -4.918 -12.053 ...
 $ z     : num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
```

## Compilación

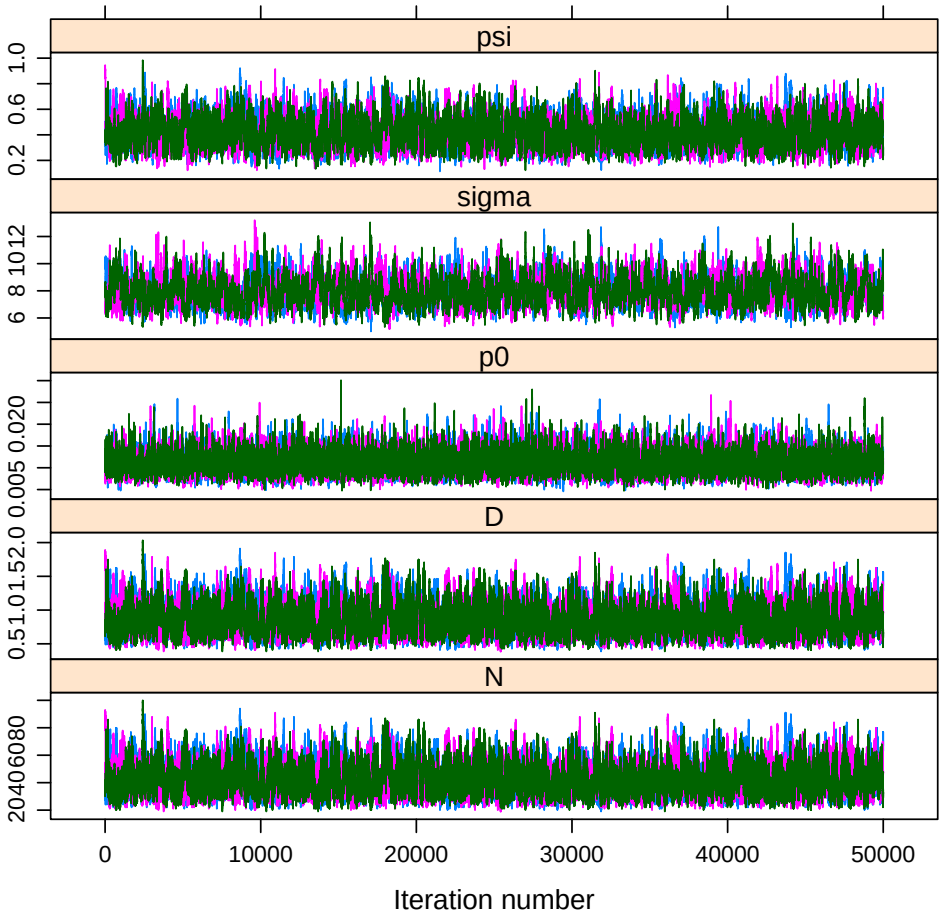
```
> Rmodel <- nimbleModel(code=code, constants=constants, data=data,
+                       inits=inits, check=FALSE, calculate=FALSE)
> Rmodel$initializeInfo()
> model <- compileNimble(Rmodel)
```

```
> params<- c('N', 'D', 'sigma','psi', 'p0','alpha0','alpha1','s','z')
> mcmc<-configureMCMC(Rmodel, monitors=params)

===== Monitors =====
thin = 1: N, D, sigma, psi, p0, alpha0, alpha1, s, z
===== Samplers =====
RW sampler (203)
  - alpha0
  - alpha1
  - psi
  - s[]  (200 elements)
binary sampler (100)
  - z[]  (100 elements)

> SCRMCMC <- buildMCMC(mcmc)
> CSCRMCMC <- compileNimble(SCRMCMC, project = Rmodel)
> # Ejecutamos el modelo
> nb=10000      # Iteraciones a desechar
> ni=50000 +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 8.918362 mins

> summary(outNim[,c('N','D','p0','psi', 'sigma')])

Iterations = 1:50000
```

```
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

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

          Mean        SD  Naive SE Time-series SE
N     42.0563 10.441966 2.696e-02      0.2187491
D      0.8554  0.212374 5.483e-04      0.0044490
p0     0.0111  0.002455 6.339e-06      0.0000284
psi    0.4224  0.113062 2.919e-04      0.0022828
sigma  7.9875  1.046099 2.701e-03      0.0254356

2. Quantiles for each variable:

           2.5%       25%      50%      75%    97.5%
N     26.000000 34.000000 41.00000 48.00000 66.00000
D      0.528802  0.691511  0.83388  0.97625  1.34234
p0     0.007123  0.009358  0.01083  0.01253  0.01675
psi    0.235928  0.340834  0.41068  0.49159  0.67741
sigma  6.233575  7.239587  7.89471  8.63689 10.28197
```

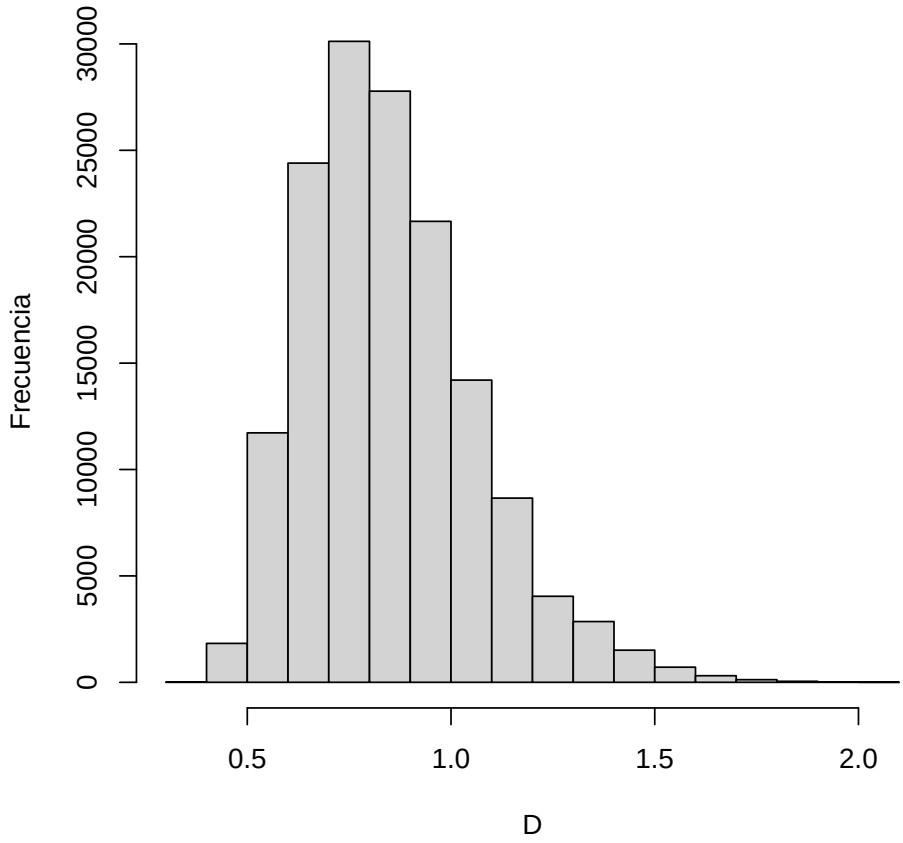
# Inspección de resultados

```
> xyplot(outNim[, c('N','D', 'p0','sigma','psi')])
> gelman.diag(outNim[,c('N','D','p0','psi', 'sigma')], multivariate = FALSE)

Potential scale reduction factors:

      Point est. Upper C.I.
N              1       1.00
D              1       1.00
p0             1       1.00
psi            1       1.01
sigma          1       1.01
```




Veamos el histograma, a ver si el aumentado de datos era suficiente

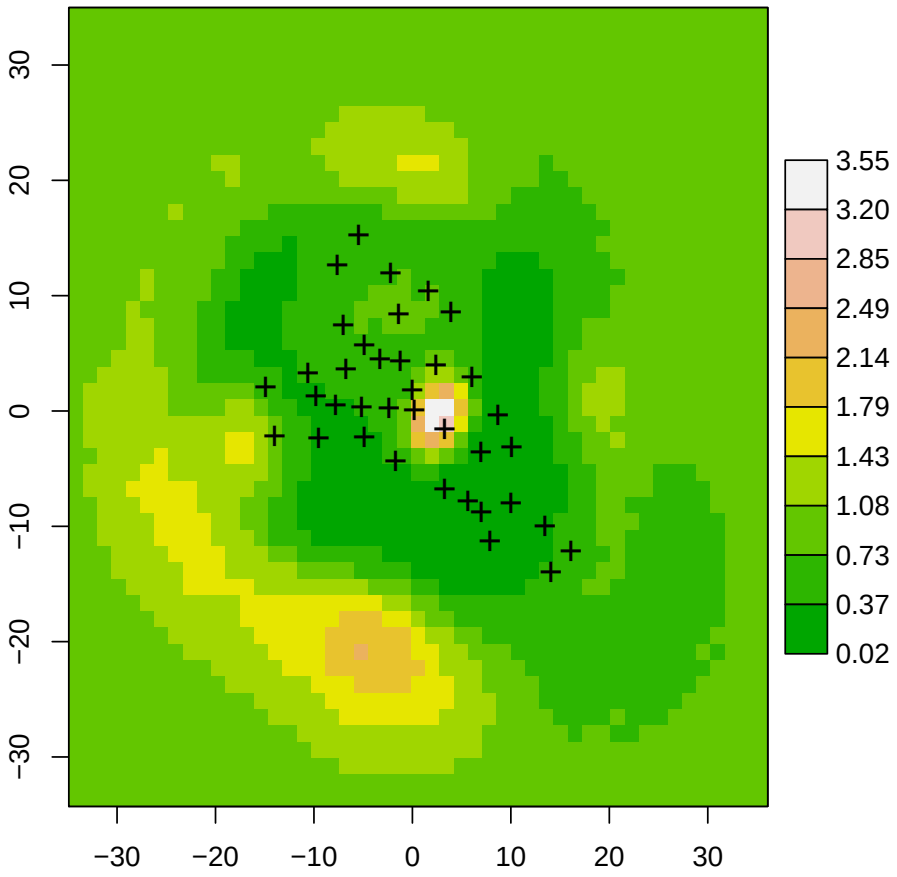
```
> hist(as.matrix(outNim[,1]), main="", xlab="D", ylab="Frecuencia")
```




# Ploteado de centros de actividad

```
> samplesn<-as.matrix(outNim)
> s1 <- samplesn[,c(7:106)]   ; Sx<-as.matrix(s1)
> s2 <- samplesn[,c(107:206)] ; Sy<-as.matrix(s2)
> z <-  samplesn[,c(208:307)] ; z<- as.matrix(z)
> delta<-20
> Xl<-min(X[,1])-delta
> Xu<-max(X[,1])+delta
```

```
> Yl<-min(X[,2])-delta
> Yu<-max(X[,2])+delta
> obj<-list(Sx=Sx,Sy=Sy,z=z)
> JJ<-SCRdensity(obj, nx=50, ny=50, scalein=1, scaleout = 100,
+                 Xl=Xl, Xu=Xu, Yl=Yl, Yu=Yu)

mean:  0.8553633

> points(X, pch="+", col=1, lwd=1, cex=1.5)
```



# Discusión del muestreo

Sollmann et al. (2012) y Sun et al. (2014) demostraron que una distancia óptimo entre trampas para los métodos SCR debería estar entre $\sigma$ y $2 * \sigma$. Así, de acuerdo a nuestro resultado, la distancia debiera estar entre 8-16 km. En nuestro caso estaba entre 1.6-4.3 km. Esto, *per se* no es un problema, pero no hacía falta una red tan densa. POr contra, si podría haber algún problema con el tamaño de malla. Lo podemos calcular con MCP del paquete `adehabitatHR` (Calenge,2006):

```
> library(adehabitatHR)
> X2<-read.table("traps.txt", header=TRUE)
> xy1<-cbind(X2[,2], X2[,3])
> xy1<-data.matrix(xy1)
> xysp1 <- SpatialPoints(xy1)
> mcp(xysp1)
```

```
Object of class "SpatialPolygonsDataFrame" (package sp):

Number of SpatialPolygons:  1

Variables measured:
  id     area
a  a 39838.18
```

Se muestreó sobre 39000 has. Sin embargo, de acuerdo a Sollmann et al. (2012) y Sun et al. (2014) el ámbito mínimo a muestrear es de al menos el de un área de campeo, para poder captar el el movimiento completo de un animal. De acuerdo a fuentes bibliográficas, en el leopardo de las nieves es de 13000 para las hembras; 22000 para los machos y hasta 50000 en sitios poco productivos. A partir de estas cifras, y observando el spiderplot, es posible que el área de estudio esté por debajo de la superficie que se precisa.

```
> sigma<-8
> r.95 <- sigma * sqrt(5.99)      # radio de HR Kernel 95%
> A.95 <- 3.14 *(r.95)^2          # area  de HR Kernel 95%
> A.95*100
```

```
[1] 120375
```

Aquí estamos obteniendo un área de campeo de 120375 has, que es más del doble de lo descrito hasta ahora. Es posible 1) que el animal registrado sea un dispersante, 2) que los

leopardos de las nieves tengan áreas de campeo móviles o Markovianas, o 3) que las áreas de campeo conocidas no fueran representativas de nuestro caso. En los dos primeros casos, sería fácil hacer una estima con esa particularidad, haciendo que las localizaciones de los animales dependan de la ocasión, y de la anterior ubicación. Como podeis ver este trabajo a posteriori sobre nuestra estima resulta el más interesante, por los cuestiones de índole biológica que nos podemos plantear a partir de algo que escapa a nuestra percepción si no usamos SCR.

# REFERENCIAS

Calenge, C. 2006. The Package Adehabitat for the R Software: Tool for the Analysis of Space and Habitat Use by Animals. *Ecological Modelling* 197: 1035.

Kachel, Shannon. 2016. More Than One Way to Count a Cat: Structured Camera-Trapping Vs. Opportunistic Fecal Sampling for Spatial Capture-Recapture Estimation of Snow Leopard (*Panthera uncia*) Population Density. *Open Science Framework*. doi:10.17605/ OSF.IO/ HR6XM.

R Core Team. 2017. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Royle, Andy, Richard Chandler, Rahel Sollmann, and Beth Gardner. 2017. Scrbook: Companion to the Book: Spatial Capture Recapture (2014).

Royle, J. Andrew, Richard B. Chandler, Rahel Sollmann, and Beth Gardner, eds. 2014. *Spatial Capture-Recapture*. Boston: Academic Press. doi:https://doi.org/10.1016/B978-0-12-405939-9.00021-9.

Sollmann, Rahel, Beth Gardner, and Jerrold L. Belant. 2012."How Does Spatial Study Design Influence Density Estimates from Spatial Capture-Recapture Models. *PLoS ONE*, no. 4: 1–8. doi:10.1371/journal.pone.0034575.