SEGUIMIENTO DE LA DIVERSIDAD BIOLÓGICA

Modelos de ocupación

José Jiménez García-Herrera (IREC-CSIC)

Universidad de Castilla-La Mancha

Los modelos de ocupación describen presencia/ausencia de una especie en un territorio. Son muy útiles como descriptores de correlaciones y procesos ecológicos. También se usan como alternativas baratas y sencillas de la abundancia. Se basan en el uso de dos procesos binomiales anidados: la especie está o no está en un sitio, y caso de estar (condicional), se detecta o no en los muestreos. La estima se infiere a partir de réplicas espaciales (sitios de muestreo) y temporales (ocasiones de muestreo), entre las cuales las poblaciones deben ser cerradas (sin entradas o salidas geográficas o demográficas). Los modelos de ocupación pueden abiertos, al añadirles una componente temporal (modelos dinámicos o multiestación). Se han desarrollado además modelos multiestado, multimétodo, multiespecie, etc.

Repetid los análisis que se ejecutan aquí con `R` (R Core Team, 2020), cambiando cada uno de vosotros el valor de *rnd* que controla la aleatoriedad de la simulación. Pegad los resultados en un procesador de texto (valores que habeis seleccionado, resultados numéricos y gráficos) y enviádmelo por e-mail a: Jose.Jimenez@csic.es.

Los scripts que se usan aquí están extraídos del libro de Kéry, M., & Schaub, M. (2012). Bayesian population analysis using WinBUGS. A hierarchical perspective. Bayesian Population Analysis using WinBUGS. Academic Press / Elsevier. http://doi.org/10.1016/B978-0-12-387020-9.00014-6.

1. Modelo sin covariables

Simulación de datos

En los procesos de modelado, emplear simulaciones nos va a permitir comparar datos "perfectos" (no afectados por errores o heterogeneidad) con los resultados de los modelos usando esos datos. Por otro lado las simulaciones también, en una segunda instancia, nos van a servir para preparar el trabajo de campo y prever el tamaño de muestra a partir de un error máximo deseado. Por último, las simulaciones son muy adecuadas para el aprendizaje. Elegimos

el tamaño de muestra y preparamos la matriz para contener los datos observados.

```
> set.seed(24)
> M <- 100                                # Número de sitios
> J <- 4                                  # repeticiones temporales
> y <- matrix(NA, nrow = M, ncol = J)  # Matriz para contener los datos
>                                         # observados
```

Valores de los parámetros

```
> psi <- 0.6          # Probabilidad de ocupación o presencia
> p <- 0.4            # Probabilidad de detección
```

Proceso ecológico

Lo que hay. Generamos datos de presencia/ausencia de la especie objetivo.

```
> z <- rbinom(n = M, size = 1, prob = psi)  # R no tiene Bernoulli
```

Proceso de observación

Lo que vemos. Generamos datos de detección/no detección.

```
> for(j in 1:J){
+    y[,j] <- rbinom(n = M, size = 1, prob = z*p)
+ }
```

Veamos los datos

```
> sum(z)                  # Sitios realmente ocupados
```

```
[1] 67
```

```
> sum(apply(y, 1, max))   # Sitios que se observa la ocupación
```

```
[1] 59
```

```
> head(cbind(z=z, y))     # Realidad y observación para los sitios 1:6
```

```
     z
[1,] 1 1 1 0 0
[2,] 1 1 1 0 0
[3,] 0 0 0 0 0
```

```
[4,] 1 0 0 0 0
[5,] 0 0 0 0 0
[6,] 0 0 0 0 0
```

Modelo en una aproximación bayesiana (usando JAGS)

Preparamos los datos para el análisis

```
> str( win.data <- list(y = y, M = nrow(y), J = ncol(y)) )
```

```
List of 3
 $ y: int [1:100, 1:4] 1 1 0 0 0 0 0 0 0 1 ...
 $ M: int 100
 $ J: int 4
```

Especificamos el modelo en BUGS

Aqui vamos a usar `JAGS` (Plummer, 2003) con `jagsUI` (Kellner, 2020).

```
> cat(file = "model.txt",
+ "
+ model {
+    # Información a priori
+    psi ~ dunif(0, 1)
+    p ~ dunif(0, 1)
+    # Probabilidad
+    for (i in 1:M) {              # Bucle sobre los sitios
+       z[i] ~ dbern(psi)          # Proceso ecológico
+       for (j in 1:J) {           # Bucle sobre los muestreos replicados
+          y[i,j] ~ dbern(z[i]*p)  # Proceso de observación
+       }
+    }
+ }
+ ")
```

Valores de inicio

```
> zst <- apply(y, 1, max)   # Para evitar conflictos
>                           # entre datos/modelo/inicio
```

```
> inits <- function(){list(z = zst)}
```

Parámetros a monitorizar

```
> params <- c("psi", "p")
```

Configuración MCMC

```
> ni <- 5000   ;   nt <- 1   ;   nb <- 1000   ;   nc <- 3
```

Ejecución del modelo

```
> library(jagsUI)
> fm2 <- jags(win.data, inits, params, "model.txt", n.chains = nc,
+    n.thin = nt, n.iter = ni, n.burnin = nb)
Processing function input.......

Done.

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 400
   Unobserved stochastic nodes: 102
   Total graph size: 606

Initializing model

Adaptive phase.....
Adaptive phase complete


 Burn-in phase, 1000 iterations x 3 chains


Sampling from joint posterior, 4000 iterations x 3 chains
```

```
Calculating statistics.......

Done.
```

```
> print(fm2, dig = 3)
```

```
JAGS output for model 'model.txt', generated by jagsUI.
Estimates based on 3 chains of 5000 iterations,
adaptation = 100 iterations (sufficient),
burn-in = 1000 iterations and thin rate = 1,
yielding 12000 total samples from the joint posterior.
MCMC ran for 0.06 minutes at time 2020-12-28 23:08:49.

            mean     sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
psi        0.696  0.065   0.574   0.695   0.825    FALSE 1 1.002  1770
p          0.378  0.038   0.304   0.378   0.452    FALSE 1 1.002   970
deviance 370.445 17.578 340.729 368.611 408.286    FALSE 1 1.003  1134

Successful convergence based on Rhat values (all < 1.1).
Rhat is the potential scale reduction factor (at convergence, Rhat=1).
For each parameter, n.eff is a crude measure of effective sample size.

overlap0 checks if 0 falls in the parameter's 95% credible interval.
f is the proportion of the posterior with the same sign as the mean;
i.e., our confidence that the parameter is positive or negative.

DIC info: (pD = var(deviance)/2)
pD = 154.2 and DIC = 524.694
DIC is an estimate of expected predictive error (lower is better).
```

Modelo en una aproximación frecuentista (usando *unmarked*)

Vamos a usar los mismos datos con `unmarked` (Fiske and Chandler, 2011), y comparar los resultados:

```
> library(unmarked)                  # Cargamos la librería
> umf <- unmarkedFrameOccu(y = y)    # preparamos datos
> summary(umf)                       # Resumen de datos
```

```
unmarkedFrame Object

100 sites
Maximum number of observations per site: 4
Mean number of observations per site: 4
Sites with at least one detection: 59

Tabulation of y observations:
  0   1
295 105
```

```
> (fm <- occu(~1 ~1, umf))          # Ejecutamos el modelo
```

```
Call:
occu(formula = ~1 ~ 1, data = umf)

Occupancy:
 Estimate    SE    z P(>|z|)
    0.816 0.307 2.66 0.00776

Detection:
 Estimate    SE     z P(>|z|)
   -0.496 0.164 -3.03 0.00245

AIC: 448.9412
```

```
> # En escala real
> backTransform(fm, type="state")
```

```
Backtransformed linear combination(s) of Occupancy estimate(s)

 Estimate     SE LinComb (Intercept)
    0.693 0.0652   0.816           1

Transformation: logistic
```

```
> backTransform(fm, type="det")
```

```
Backtransformed linear combination(s) of Detection estimate(s)
```

```
 Estimate     SE LinComb (Intercept)
    0.379 0.0385  -0.496           1

Transformation: logistic
```

2. Modelo con covariables

Simulación de datos

Generamos datos

```
> set.seed(1)
> M <- 100              # Número de sitios
> J <- 3                # Número de presencias/ausencias
> y <- matrix(NA, nrow = M, ncol = J) # para contener los datos
>                                     # de observaciones
```

Creamos una covariable para la ocupación de "altura de la vegetación" y la llamamos vegHt:

```
> vegHt <- sort(runif(M, -1, 1)) # lo ordenamos porque nos conviene
>                                # para los gráficos
```

Elegimos valores de parámetros para el modelo

```
> beta0 <- 0            # Intercepto en escala logit
> beta1 <- 3            # Pendiente en escala logit para vegHt
> psi <- plogis(beta0 + beta1 * vegHt) # Probabilidad de ocupación
```

Simulamos para cada sitio y las verdaderas presencias/ausencias

```
> z <- rbinom(M, 1, psi)        # Verdadera presencia/ausencia
```

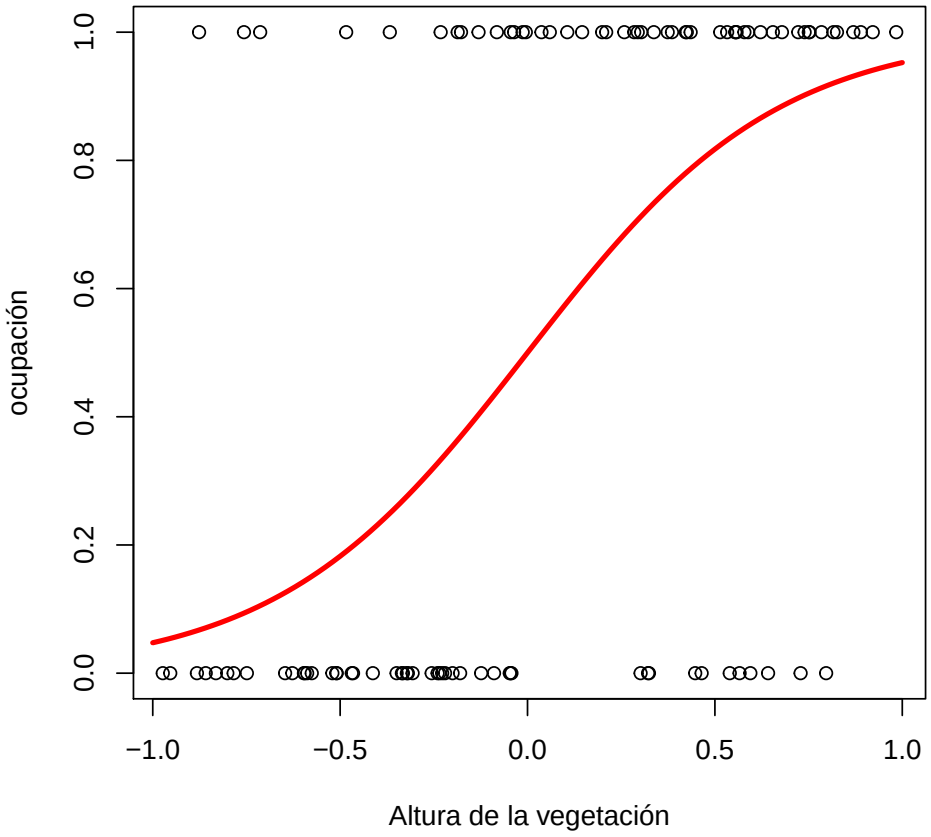
Vemos los datos reales que hemos creado

```
> table(z)
```

```
z
 0  1
49 51
```

Ploteamos el verdadero estado (relación entre la covariable y la ocupación)

```
> plot(vegHt, z, xlab="Altura de la vegetación", ylab="ocupación")
> plot(function(x) plogis(beta0 + beta1*x), -1, 1, add=T, lwd=3, col = "red")
```



Vamos ahora a simular la detección, creando una covariable para ella. Será el "viento", para el que suponemos que va a disminuir la detección con la velocidad del viento.

```
> viento <- array(runif(M * J, -1, 1), dim = c(M, J))
> alpha0 <- -2                          # intercepto en escala logit
> alpha1 <- -3                          # Pendiente en escala logit para viento
> p <- plogis(alpha0 + alpha1 * viento) # Probabilidad de detección
```

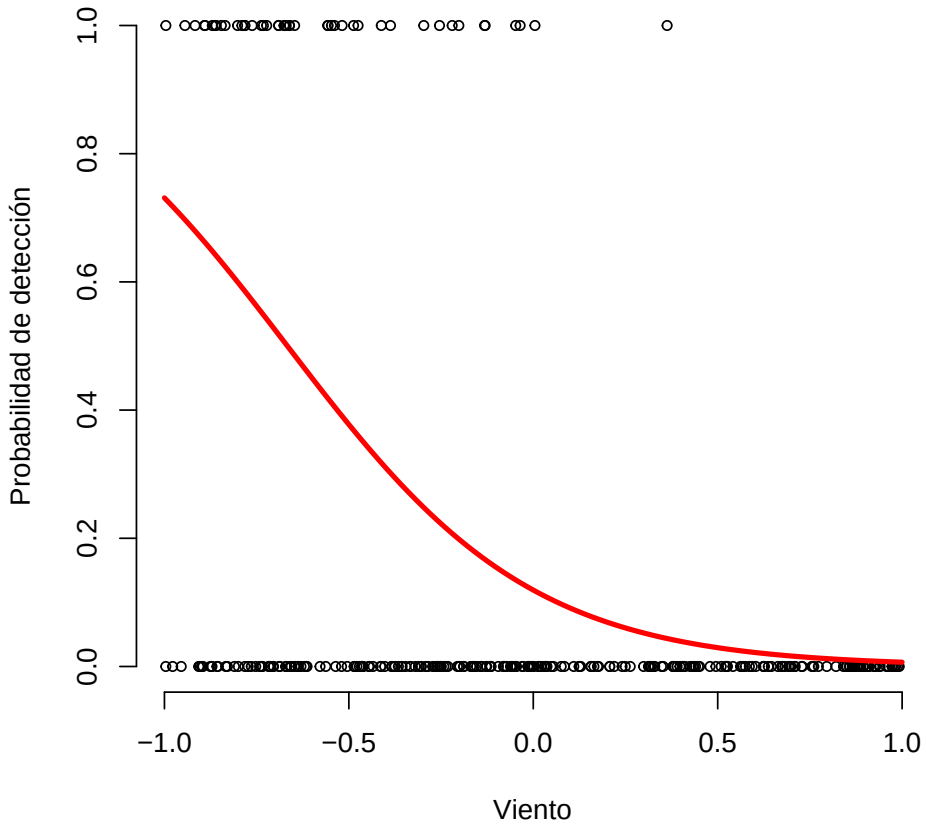
Simulamos ahora la detección con esta covariable. Realizamos J = 3 réplicas del muestreo en cada sitio

```
> for(j in 1:J) {
```

```
+     y[,j] <- rbinom(M, z, p[,j])
+ }
> sum(apply(y, 1, max))      # Número de sitios con presencias observadas
[1] 32
```

Ploteamos los datos observados y efectos del viento en la probabilidad de detección

```
> plot(viento, y, xlab="Viento", ylab="Probabilidad de detección",
+   frame = F, cex = 0.75)
> plot(function(x) plogis(alpha0 + alpha1*x), -1, 1, add=T, lwd=3, col = "red")
```



Inspeccionamos los datos: ocupación real y lo que percibimos (y)

```
> head(cbind(psi=round(psi,2), z=z, y1=y[,1], y2=y[,2], y3=y[,3]))
```

```
      psi z y1 y2 y3
[1,] 0.05 0  0  0  0
[2,] 0.05 0  0  0  0
[3,] 0.07 0  0  0  0
[4,] 0.07 1  0  0  0
[5,] 0.07 0  0  0  0
[6,] 0.08 0  0  0  0
```

Modelo en una aproximación bayesiana usando NIMBLE

En esta ocasión, en vez de JAGS vamos a utilizar NIMBLE (https://r-nimble.org/) (de Valpine et al., 2017; de Valpine et al., 2020) que es un software con una extraordinaria potencialidad. Al igual que el modelo anterior, el modelo con covariable también se puede ejecutar con `unmarked`.

Preparamos datos y constantes:

```
> str(data    <- list(y = y, vegHt = vegHt, viento = viento))
```

```
List of 3
 $ y     : int [1:100, 1:3] 0 0 0 0 0 0 0 0 1 0 ...
 $ vegHt : num [1:100] -0.973 -0.953 -0.882 -0.876 -0.859 ...
 $ viento: num [1:100, 1:3] -0.465 -0.5627 0.0336 -0.4621 -0.6377 ...
```

```
> str(constants<-list(M = nrow(y), J = ncol(y),
+                     XvegHt =  seq(-1, 1, length.out=100),
+                     Xviento = seq(-1, 1, length.out=100)))
```

```
List of 4
 $ M      : int 100
 $ J      : int 3
 $ XvegHt : num [1:100] -1 -0.98 -0.96 -0.939 -0.919 ...
 $ Xviento: num [1:100] -1 -0.98 -0.96 -0.939 -0.919 ...
```

```
> library(nimble)
> code <- nimbleCode({
+
+   # A priori
+   mean.p ~ dunif(0, 1)      # prior del intercepto de deteccion
+   alpha0 <- logit(mean.p)  # Intercepto de deteccion
+   alpha1 ~ dunif(-20, 20)  # Prior de la covariable deteccion **viento**
+   mean.psi ~ dunif(0, 1)   # distribución de la ocupacion
+   beta0 <- logit(mean.psi) # Prior del intercepto de ocupación
+   beta1 ~ dunif(-20, 20)   # Prior para la covariable ocupación **vegHt**
+
+   # Probabilidad
+   for (i in 1:M) {
+     # Modelo de estado o proceso ecológico
+     z[i] ~ dbern(psi[i])      # Verdadera ocupacion (z) en el sitio i
+     logit(psi[i]) <- beta0 + beta1 * vegHt[i]
+     for (j in 1:J) {
+       # Modelo de observacion para la observacion real
+       y[i,j] ~ dbern(p.eff[i,j])    # deteccion-no deteccion en i y j
+       p.eff[i,j] <- z[i] * p[i,j]
+       logit(p[i,j]) <- alpha0 + alpha1 * viento[i,j]
+     }
+   }
+
+   # Cantidades derivadas
+   N.occ <- sum(z[1:M])       # Numero de sitios ocupados
+   psi.fs <- N.occ/M          # Proporcion de sitios ocupados
+   for(k in 1:100){
+     logit(psi.pred[k]) <- beta0 + beta1 * XvegHt[k] # predicciones de psi
+     logit(p.pred[k]) <- alpha0 + alpha1 * Xviento[k]  # predicciones de p
+   }
+ })
```

Valores de inicio: deben suministrarse para las mismas dimensiones y similares cantidades que los *priori* dados

```
> zst <- apply(y, 1, max)
> str(inits <- list(z = zst, mean.p = runif(1), alpha1 = runif(1),
```

```
+                mean.psi = runif(1), beta1 = runif(1)))
List of 5
 $ z       : int [1:100] 0 0 0 0 0 0 0 0 1 0 ...
 $ mean.p  : num 0.392
 $ alpha1  : num 0.568
 $ mean.psi: num 0.0952
 $ beta1   : num 0.194
```

Parámetros a monitorizar

```
> params <- c('alpha0', 'alpha1', 'beta0', 'beta1', 'N.occ', 'psi.fs',
+   'psi.pred', 'p.pred', 'z')
```

Ejecución del modelo en Nimble

```
> Rmodel <- nimbleModel(code=code, constants=constants, data=data,
+                       inits=inits, check=FALSE, calculate=FALSE)
> Cmodel <- compileNimble(Rmodel)
> mcmcspec<-configureMCMC(Rmodel, monitors=params, nthin=10)
===== Monitors =====
thin = 1: alpha0, alpha1, beta0, beta1, N.occ, psi.fs, psi.pred, p.pred, z
===== Samplers =====
RW sampler (4)
  - mean.p
  - alpha1
  - mean.psi
  - beta1
binary sampler (100)
  - z[]  (100 elements)
> pumpMCMC <- buildMCMC(mcmcspec)
> CpumpMCMC <- compileNimble(pumpMCMC, project = Rmodel)
> ## Ejecutamos con estas características
> nb=10000
> ni=25000+nb
> nc=3
> start.time<-Sys.time()
> outNim <- runMCMC(CpumpMCMC, niter = ni , nburnin = nb ,
+                   nchains = nc, inits=inits,
```

```
+                    setSeed = TRUE, progressBar = TRUE,
+                    samplesAsCodaMCMC = TRUE)
|-------------|-------------|-------------|-------------|
|-----------------------------------------------------|
|-------------|-------------|-------------|-------------|
|-----------------------------------------------------|
|-------------|-------------|-------------|-------------|
|-----------------------------------------------------|
> end.time<-Sys.time()
> end.time-start.time
Time difference of 30.13033 secs
```

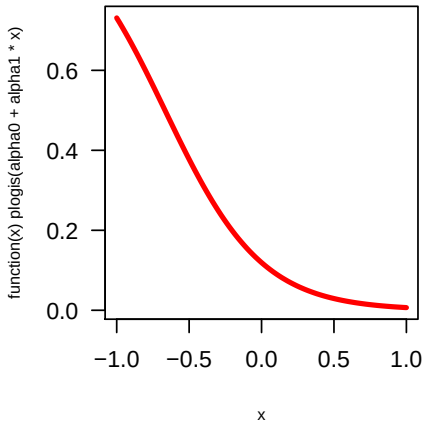
Comparamos la realidad con la inferencia bayesiana en la tabla:

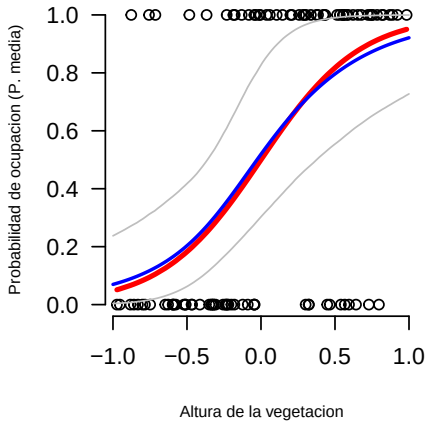
```
> realidad <- c('alpha0'=alpha0, 'alpha1'=alpha1, 'beta0'=beta0,
+               'beta1'=beta1, 'N.occ'=sum(z), 'psi.fs'=sum(z)/M)
> samples<-rbind(as.matrix(outNim$chain1),
+                as.matrix(outNim$chain2),
+                as.matrix(outNim$chain2))
> ref0<-apply(samples[,2:5],2,mean)
> ref1<-sum(apply(samples[,207:306],2,mean))
> print(cbind('realidad'=realidad, 'estimado'=c(ref0, ref1, ref1/M)))
       realidad    estimado
alpha0    -2.00 -1.74866452
alpha1    -3.00 -3.04771926
beta0      0.00  0.09958541
beta1      3.00  3.13230578
N.occ     51.00 52.19468000
psi.fs     0.51  0.52194680
```

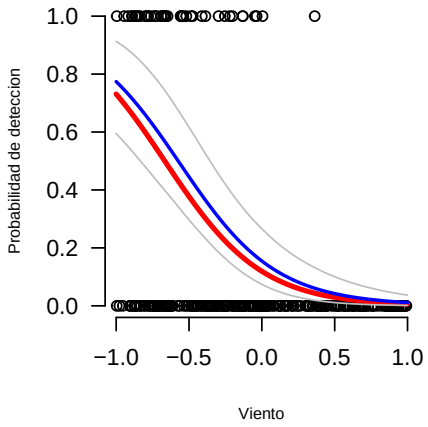
Ploteado

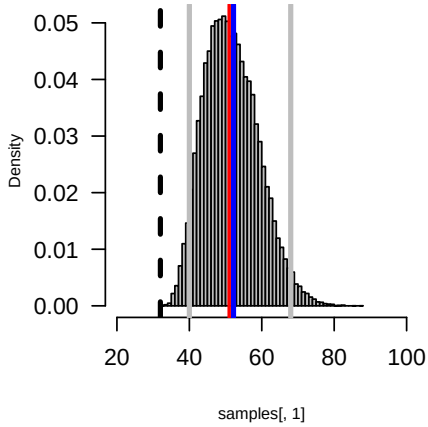
```
> par(mfrow = c(2, 2), mar = c(5, 5, 2, 2), las = 1,
+   cex.lab = 0.75, cex = 0.8)
> plot(function(x) plogis(alpha0 + alpha1*x), -1, 1,
+   lwd=3, col = "red")
> plot(vegHt, z, xlab="Altura de la vegetacion",
```

```
+   ylab="Probabilidad de ocupacion (P. media)",
+   las = 1, frame = F)   # Verdadera presencia/ausencia
> lines(vegHt, psi, lwd=3, col="red")   # psi real
> lines(constants$XvegHt, apply(samples[,107:206],2,mean),
+   col="blue", lwd = 2)
> lower.psi<-apply(samples[,107:206],2,quantile, 0.025)
> upper.psi<-apply(samples[,107:206],2,quantile, 0.975)
> matlines(constants$XvegHt, cbind(lower.psi,upper.psi),
+          col="grey", lty = 1)
> plot(viento, y, xlab="Viento",
+      ylab="Probabilidad de deteccion", frame = F)
> plot(function(x) plogis(alpha0 + alpha1*x), -1, 1,
+      add=T, lwd=3, col = "red")
> lines(constants$Xviento, apply(samples[,6:105],2,mean),
+       col="blue", lwd = 2)
> lower.p<-apply(samples[,6:105],2,quantile, 0.025)
> upper.p<-apply(samples[,6:105],2,quantile, 0.975)
> matlines(constants$Xviento, cbind(lower.p,upper.p),
+          col="grey", lty = 1)
> # Ploteado de posteriores del numero de sitios ocupados
> hist(samples[,1], col = "grey", breaks = 60, xlim = c(20, 100),
+ main = "", freq = F)
> abline(v = mean(samples[,1]), col = "blue", lwd = 3)
> lower.Nocc<-quantile(samples[,1],0.025)
> upper.Nocc<-quantile(samples[,1],0.975)
> abline(v = c(lower.Nocc,upper.Nocc), col = "grey", lwd = 3)
> abline(v = sum(apply(y, 1, max)), lty = 2, lwd = 3)
> abline(v = sum(z), col = "red", lwd = 2)
```


function(x) plogis(alpha0 + alpha1 * x)
0.0
0.2
0.4
0.6
−1.0
−0.5
0.0
0.5
1.0
x


Probabilidad de ocupacion (P. media)
0.0
0.2
0.4
0.6
0.8
1.0
−1.0
−0.5
0.0
0.5
1.0
Altura de la vegetacion


Probabilidad de deteccion
0.0
0.2
0.4
0.6
0.8
1.0
−1.0
−0.5
0.0
0.5
1.0
Viento


Density
0.00
0.01
0.02
0.03
0.04
0.05
20
40
60
80
100
samples[, 1]

3. REFERENCIAS

de Valpine, P., Paciorek, C. J., Turek, D., Michaud, N., Anderson-Bergman, C., Obermeyer, F., ... Bodik, R. (2017). Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, 26(2), 403–413. doi:10.1080/10618600.2016.1172487.

de Valpine, P., Paciorek, C. J., Turek, D., Michaud, N., Anderson-Bergman, C., Obermeyer, F., ... Paganin, S. (2020). NIMBLE: MCMC, Particle Filtering, and Programmable Hierarchical Modeling. doi:10.5281/zenodo.1211190.

Fiske, I. J., & Chandler, R. B. (2011). `unmarked`: An R package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software*, 43(10), 1–23. doi:10.1002/wics.10.

Kellner, K. (2020). jagsUI: A Wrapper Around 'rjags' to Streamline 'JAGS' Analyses. R package version 1.5.1.9100. Retrieved from https://github.com/kenkellner/jagsUI.

Kéry, M., & Schaub, M. (2012). *Bayesian population analysis using WinBUGS. A hierarchical perspective*. Academic Press / Elsevier. doi:10.1016/B978-0-12-387020-9.00014-6.

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. 3rd International Workshop on Distributed Statistical Computing (DSC 2003). Vienna, Austria. Retrieved from http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/

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/.