Apéndice 1: La unión hace la fuerza: modelos de distribución de especies integrando diferentes fuentes de datos

Javier Fernández-López (javfer05@ucm.es), Pelayo Acevedo, Olivier Gimenez

21-10-2022

En este documento se reproducen los análisis mostrados en el trabajo *"La unión hace la fuerza: modelos de distribución de especies integrando diferentes fuentes de datos"* para la revista Ecosistemas.

Simulación de abundancia

Comenzamos simulando la distribución de una serie de individuos en nuestro área de estudio, siguiendo un Proceso de Puntos de Poisson no homogéneo. El número de individuos en cada localidad i (N_i) vendrá determinado por:

$$N_i \sim Poisson(\lambda_i)$$

Como nuestro Proceso de Poisson es "no homogéneo", la intensidad de puntos (individuos) en cada localización i (λ_i) dependerá a su vez de dos variables ambientales, la altitud (alt) y la cobertura forestal (bosque), siguiendo:

$$log(\lambda_i) = \beta_0 + \beta_1 * alt_i + \beta_2 * bosque_i$$

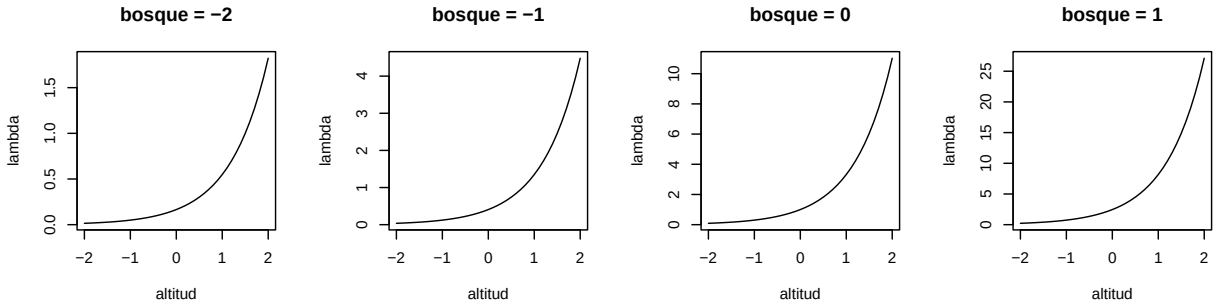
Los valores que hemos elegido para cada uno de los coeficientes son:

- $\beta_0 = 0$ (nuestro modelo tendrá intercepto = 0 para simplificar)
- $\beta_1 = 1.2$ (a nuestra especie le afecta positivamente la altitud)
- $\beta_2 = 0.9$ (a nuestra especie le afecta positivamente la cobertura forestal)

Aunque el Proceso de Puntos de Poisson sucede en un espacio continuo, es necesario suministrar a la función `rpoispp()` un `raster` que describa la distribución de la intensidad de puntos (λ) en todo el territorio con una resolución determinada. Por tanto, en este ejemplo asumiremos que todos los procesos suceden a la mayor resolución posible (a nivel de celda i).

```
library(raster)
library(ggplot2)
library(dismo)
library(rgeos)
library(spatstat)
library(nimble)
library(coda)
set.seed(366)

# Simulación de covariables.  Altitud
alt <- raster(nrows = 100, ncols = 100, xmn = 0, xmx = 100, ymn = 0, ymx = 100)
alt <- -scale(distanceFromPoints(alt, c(35, 35)))
```

```r
# Bosque
bosque <- raster(nrows = 10, ncols = 10, xmn = 0, xmx = 100, ymn = 0, ymx = 100)
bosque[] <- runif(100, 1, 10)
bosque <- scale(disaggregate(bosque, 10, "bilinear"))

# Distancia a carreteras (lo usaremos luego).
vv <- randomPoints(alt, 15)
carret <- voronoi(vv, ext = extent(alt) + 100)
dd <- gDistance(as(carret, "SpatialLines"), as(alt, "SpatialPoints"), byid = TRUE)
dcarret <- alt
dcarret[] <- scale(apply(dd, 1, min))

# Simulación de un Proceso de Puntos de Poisson no homogéneo.
# Función para calcular lambda. Beta1 = 1.2; Beta2 = 0.9.  Al utilizar la
# función vínculo 'log' debemos tomar exp() al otro lado de la ecuación
ff <- function(x1, x2) {
    exp(1.2 * (x1) + 0.9 * (x2))
}
rotate <- function(x) (apply(x, 2, rev))

# Creamos lambda y simulamos el proceso de puntos (individuos).
Z <- as.im(rotate(matrix(ff(alt[], bosque[]), 100, 100, byrow = TRUE)),
    W = owin(xrange = c(0, 100), yrange = c(0, 100)))
pp <- rpoispp(Z)  # Proceso de Puntos de Poisson no homogéneo.

# Almacenamos lambda en un ráster.
rZ <- raster(Z)

# Obtenemos las coordenadas de cada individuo.
ind <- data.frame(x = pp$x, y = pp$y, det = NA)

# Contamos el número de puntos en cada celda para crear un ráster de abundancias.
Q100 <- quadratcount(pp, nx = 100, ny = 100)
abun <- rZ
abun[] <- Q100
```



Simulación de dos fuentes de datos

Ya tenemos disponible nuestro área de estudio con nuestros individuos distribuidos en función de la altitud y de la cobertura forestal. Ahora simularemos dos fuentes de datos a partir de los cuales intentaremos modelizar

la abundancia de individuos de nuestra especie virtual. La primera será un muestreo llevado a cabo por investigadorxs. Consistirá en conteos directos de individuos en tan solo 10 sitios de una zona concreta del área de estudio (debido a la falta de presupuesto no se puede ampliar el muestreo), repetidos en 4 ocasiones. En situaciones reales, es muy poco probable que seamos capaces de detectar absolutamente todos los individuos, así que simularemos una detectabilidad imperfecta en cada una de esas ocasiones, por lo que el conteo de individuos que obtengamos (y_{ij}, en el sitio i y en la ocasión j) será siempre menor que el número real de individuos en cada sitio. Para ello, utilizaremos una distribución binomial en la que cada individuo de cada sitio i en cada ocasión j tendrá una probabilidad de ser detectado $p_{ij} = 0.6$, esto es, cada individuo será detectado con una probabilidad del 60% en cada intento de conteo.

$$y_{ij} \sim Binomial(N_i, p_{ij})$$

```
# Seleccionamos 10 sitios de una región concreta.
site_ID <- sample(c(cellFromPolygon(abun, carret[8, ])[[1]], cellFromPolygon(abun,
    carret[15, ])[[1]]), 10)

# Iniciamos el data.frame
conteos <- data.frame(site_ID, O1 = NA, O2 = NA, O3 = NA, O4 = NA)

# Con ayuda de un doble bucle vamos pasando en 4 ocasiones por los 10 sitios.
for (j in 1:4) {
    for (i in 1:length(site_ID)) {
        # En cada conteo, para simular una detectabilidad imperfecta
        # usaremos una distribución binomial con p = 0.6.
        conteos[i, j + 1] <- rbinom(1, extract(abun, site_ID[i]), 0.6)
    }
}

# Por último tomamos los datos de las covariables ambientales en cada uno de nuestros
# sitios de muestreo.
conteos$alt <- extract(alt, conteos$site_ID)
conteos$bosque <- extract(bosque, conteos$site_ID)

# ¡Ya tenemos listos nuestros datos de conteos repetidos!
head(conteos)
```

```
##   site_ID O1 O2 O3 O4        alt      bosque
## 1    9396  0  0  0  0 -1.3442125  0.18284910
## 2    8774  0  0  0  0 -0.1314638 -1.34028334
## 3    7272  0  0  0  0  0.2674506 -0.03017565
## 4    5769  0  0  0  0  0.4270864 -0.99779469
## 5    5782  3  1  3  4 -0.2679736  0.58094068
## 6    9235  8  8  5  5  0.7985027  1.29980969
```

La segunda fuente de datos va a consistir en registros oportunistas de la presencia de la especie provenientes de ciencia ciudadana. Este tipo de datos está a menudo afectado por un sesgo de muestreo relacionado con aquellas zonas que suelen estar más frecuentadas (como las proximidades a las carreteras o las zonas con mayor población humana), por lo que la probabilidad de registrar la especie (S_i) dependerá de una covariable: la distancia a la carretera más próxima.

$$logit(S_i) = \alpha_0 + \alpha_1 * dcarretera_i$$

En este caso, los valores que hemos elegido para cada uno de los coeficientes son:

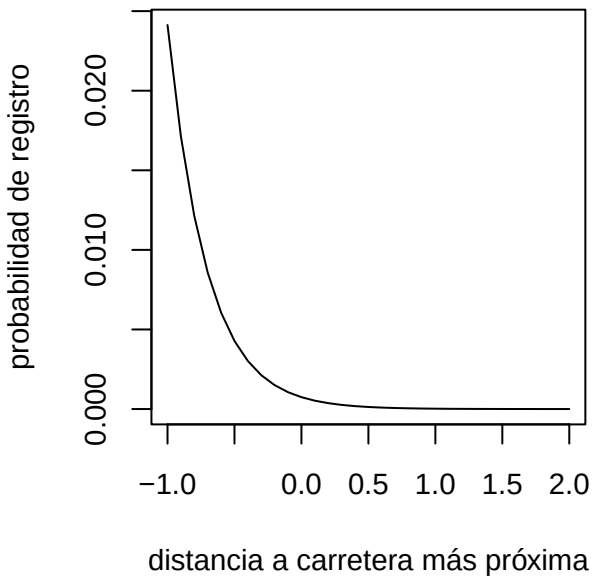
- $\alpha_0 = -7.2$ (en general, hay poca probabilidad de registrar un individuo)

- $\alpha_1 = -3.5$ (es menos probable encontrar un registro a medida que nos alejamos de la carretera)

A este tipo de datos se le suele llamar Proceso de Puntos de Poisson "adelgazado" (*thinned Poisson Point Process* Fithian et al. (2015)), y el número de puntos registrados en cada sitio i viene determinado por:

$$Nreg_i = Poisson(\lambda_i * S_i)$$

```
# Simulación de presencias oportunistas (ciencia ciudadana).

# Probabilidad de ser registrado.  Al tomar logit como función vínculo,
# tenemos que hacer el recíproco de logit al otro lado de la ecuación.
S <- exp(-7.2 - (3.5 * dcarret))/(1 + exp(-7.2 - (3.5 * dcarret)))

# Detección o no detección de cada individuo. Con este bucle pasaremos por todos los
# individuos de nuestro área de muestreo y, dependiendo de lo cerca o lejos que se
# encuentren de la carretera, serán detectados o no. Para ello, utilizaremos la
# probabilidad de ser registrado que hemos computado anteriormente (S).
for (i in 1:nrow(ind)) {
    ind$det[i] <- rbinom(1, 1, extract(S, ind[i, 1:2]))
}

# Guardamos los indivudios que SI han sido registrados.
po <- ind[ind$det == 1, 1:2]

# Creamos un data.frame con tantas filas como celdas tiene nuestro área de estudio.
# Para cada celda se pondrá un 0 si no se ha registrado ningún individuo, mientras
# que se anotará un 1 si se ha registrado al menos un individuo.
registros <- data.frame(cellID = 1:ncell(alt), det = 0)
registros$det[cellFromXY(alt, po)] <- 1
```



Resumen de la simulación

- Hemos simulado la distribución y abundancia de una población de individuos a la cual le gusta estar en mayores elevaciones y con buena cobertura forestal, siguiendo:

$$N_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = 1.2 * alt_i + 0.9 * bosque_i$$

- Además hemos simulado un set de datos de un muestreo estandarizado que consistió en 4 visitas a 10 sitios en las cuales se contaron todos los individuos que se detectaron. Lamentablemente, nuestro equipo tuvo una tasa de éxito del 60% en cada una de las visitas y además los 10 sitos se encontraron agregados en una zona concreta de nuestro área de estudio...

$$y_{ij} \sim Binomial(N_i, 0.6)$$

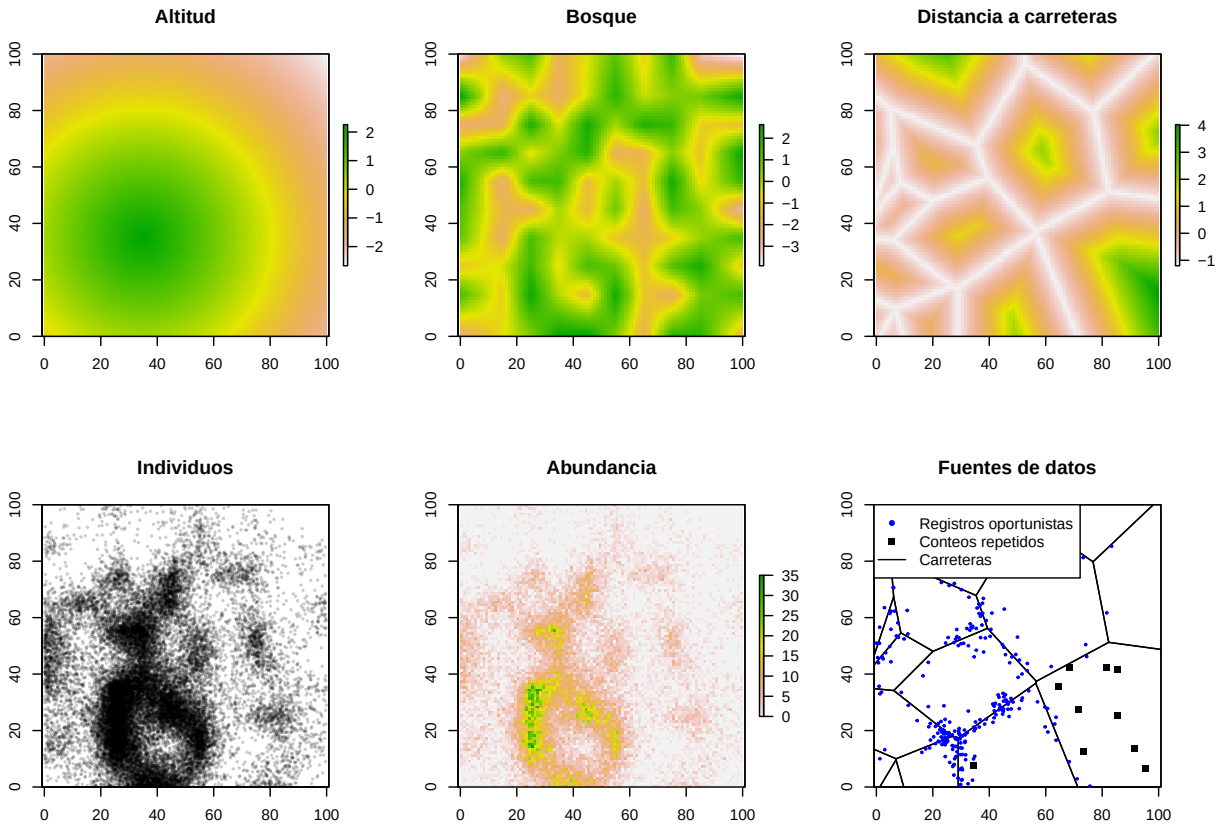
- Por último, hemos simulado un set de datos de registros de presencia oportunistas provenientes de ciencia ciudadana en todo nuestro área de estudio. Por desgracia (y como era de esperar), estos registros se acumulan en las proximidades de las carreteras, ya que es en esos lugares donde existe un mayor esfuerzo de muestreo (hay más observadores).

$$Nreg_i = Poisson(\lambda_i * S_i)$$

$$logit(S_i) = -7.2 - 3.5 * dcarret_i$$

A continuación podemos resumir de forma gráfica nuestra simulación:

```
par(mfrow = c(2, 3))
plot(alt, main = "Altitud")
plot(bosque, main = "Bosque")
plot(dcarret, main = "Distancia a carreteras")
plot(abun, alpha = 0, legend = F, main = "Individuos")
plot(pp, cex = 0.5, pch = 16, add = TRUE)
plot(abun, main = "Abundancia")
plot(abun, alpha = 0, legend = F, main = "Fuentes de datos")
lines(carret)
points(ind[ind$det == 1, 1:2], col = "blue", cex = 0.5, pch = 20)
points(xyFromCell(abun, site_ID), pch = 15, cex = 1)
legend("topleft", legend = c("Registros oportunistas", "Conteos repetidos",
    "Carreteras"), pch = c(20, 15, NA), lty = c(NA, NA, 1), col = c("blue",
    "black", "black"), bg="white")
```



Modelización de la abundancia

A continuación se incluye el código utilizado para ajustar los modelos para cada uno de los juegos de datos por separado, y finalmente el código para ajustar un modelo integrando ambas fuentes de datos mediante la metodología de *joint-likelihood*. Este ejemplo es análogo al descrito en Dorazio (2014). Por tanto, para una mayor comprensión del mismo, se recomienda su lectura. Los modelos se van a ajustar mediante inferencia Bayesiana utilizando NIMBLE (Valpine et al. 2017). El objetivo de este apéndice es únicamente mostrar un ejemplo de modelización integrando diferentes bases de datos, por lo que no se proporciona una detallada explicación sobre NIMBLE o sobre inferencia Bayesiana. Además, tampoco se incluye un análisis profundo de los resultados obtenidos con NIMBLE (exploración de los trace-plots, convergencia de las MCMC, tamaño de muestreo efectivo de las MCMC, etc.). Sin embargo, existen numerosos tutoriales y cursos gratuitos online (como **este** de Olivier Gimenez) que se recomienda consultar.

Conteos repetidos

Se ajustará un modelo N-mixture (Royle 2004) a partir de nuestros datos de conteos repetidos. La variable de estado o latente, la abundancia, viene definida como:

$$N_i \sim Poisson(\lambda_i) \quad \text{(Eq 1)}$$

$$log(\lambda_i) = \beta_1 * alt_i + \beta_2 * bosque_i \quad \text{(Eq 2)}$$

mientras que las observaciones, nuestros conteos, vendrán definidos por:

$$y_{ij} \sim Binomial(N_i, p) \quad \text{(Eq 3)}$$

En este primer modelo comentaremos el código. Los sucesivos modelos son similares, por lo que el código estará menos comentado.

```
# A continuación vamos a definir dos tipos de "inputs" para NIMBLE, las
# "constantes" y los "datos". Brevemente, las "constantes" son aquellos valores
# que no van a cambiar (como los índices definidos para ejecutar un bucle),
# mientras que los "datos" son valores que podrían cambiar (como el número de
# individuos detectados en un sitio "i")
constants <- list(nsurveys = rep(4,nrow(conteos)), # Número de ocasiones
                  nsites = nrow(conteos))          # Número de sitios

data <- list(y = as.matrix(conteos[,2:5]),  # Nuestros conteos
             alt = conteos$alt,             # Valores de alt para cada sitio
             bosque = conteos$bosque        # Valores de bosque para cada sitio
)

# Descripción del modelo
# A continuación definiremos el modelo para NIMBLE. Esta sintaxis es
# similar a las utilizadas en BUGS y JAGS.

# Primero iniciamos la estructura del modelo y lo guardamos en un objeto que
# denominaremos conteos_model
conteos_model <- nimbleCode({

    # PRIORS: A continuación definiremos los priors. Los priors pueden
    # definirse como valores probables que 'a priori' creemos que
    # podrían tomar cada uno de nuestros coeficientes.
    b_alt ~ dnorm(0, 10)  # Normal media=0 sd=10
    b_bosque ~ dnorm(0, 10)  # Normal media=0 sd=10
    p_binom ~ dbeta(1, 1)  # Beta con parámetros (1,1), flat prior entre 0 y 1

    # LIKELIHOOD: Se podría decir que este es el corazón de nuestro
    # modelo. Aquí es dónde vamos a implementar (escribir) las
    # ecuaciones que hemos descrito más arriba Comenzaremos por la
    # VARIABLE DE ESTADO O LATENTE (ABUNDANCIA)

    # En cada uno de nuestros sitios de muestreo...
    for (i in 1:nsites) {
      # El número de individuos n en la celda i sigue una distribución de Poisson
      # con lambda = "lambda en la celda i" (Eq 1)
        n[i] ~ dpois(lambda[i])
        # A su vez, esa "lambda en la celda i" se relaciona con una
        # serie de covariables predictoras siguiendo la Eq 2.
        log(lambda[i]) <- b_alt * alt[i] + b_bosque * bosque[i]
    }
    # A continuación, se definirá el likelihood para el PROCESO
    # OBSERVACIONAL, esto es, los CONTEOS REPETIDOS

    # En cada uno de nuestros sitios de muestreo...
    for (j in 1:nsites) {
       # y en cada una de las 4 ocasiones de muestreo...
        for (k in 1:nsurveys[j]) {
```

```
          # El número de individuos detectados en el sitio j y en la ocasión k
          # va a seguir una distribución binomial con parámetros 'p', y
          # N = "número de individuos que haya en la celda que estamos muestreando" (Eq 3)
            y[j, k] ~ dbin(p_binom, n[j])
        }
    }

})

# Por último, solo queda indicar una serie de valores que
# ayudarán a NIMBLE a iniciar el MCMC. No comentaremos mucho más esta
# parte, recomendamos consultar fuentes más robustas, como por
# ejemplo https://oliviergimenez.github.io/nimble-workshop/#1
inits <- function() {
    list(n = rep(1, constants$nsites), b_alt = runif(1, -1, 1), b_bosque = runif(1,
        -1, 1), p_binom = runif(1, 0, 1))
}
monitores <- c("b_alt", "b_bosque", "p_binom")

# Aquí definiremos los 'settings' para el MCMC. Como este ejemplo es
# meramente ilustrativo seleccionaremos unos valores que NO
# son los adecuados, pero ayudarán a tener los resultados de forma
# rápida. Se recomienda consultar
# https://oliviergimenez.github.io/nimble-workshop/#1
nc <- 1  # número de cadenas del MCMC (se recomiendan mínimo 3)
nb <- 5000  # iteraciones iniciales a descartar (depende, pero probablemente mas)
ni <- nb + 50000  # número de iteraciones de cada cadena (depende, pero probablemente mas)
model <- nimbleModel(code = conteos_model, data = data, constants = constants,
    inits = inits(), calculate = FALSE)
c_model <- compileNimble(model)
model_conf <- configureMCMC(model, useConjugacy = FALSE)
model_mcmc <- buildMCMC(model_conf)
c_model_mcmc <- compileNimble(model_mcmc, project = model)

# Iniciamos el MCMC
samples_conteos <- runMCMC(c_model_mcmc, nburnin = nb, niter = ni, nchains = nc)
# Al finalizar lo primero que deberíamos hacer es explorar los
# traceplots para ver cómo ha discurrido el análisis. En este ejemplo
# nos saltaremos ese paso y directamente obtendremos la media de la
# distribución a posteriori obtenida mediante el MCMC para los
# parámetros de interés y produciremos una predicción
pred_Conteos <- exp(mean(samples_conteos[, 1]) * alt + mean(samples_conteos[,
    2]) * bosque)
samples_Cont <- coda::mcmc(samples_conteos)

# Cabe destacar que una de las ventajas de la inferencia Bayesiana es
# precisamente que los coeficientes estimados no son valores fijos,
# sino distribuciones, por lo que podemos tener en cuenta la
# incertidumbre asociada. Al tomar la media de las distribuciones
# posteriores estamos 'perdiendo' mucha de la información que nos
# ofrece nuestro análisis, pero lo mantendremos de esta forma por
# simplicidad. Para explorar los traceplots podemos usar:
# coda::traceplot(samples_Cont)
```

Registros oportunistas

A continuación pasaremos a describir el modelo que utilizaremos para estimar la abundancia a partir de los registros oportunistas. Seguiremos la aproximación desarrollada por Dorazio (2014). El modelado de estos datos es un poco más complejo. Aunque el número de registros en cada una de nuestras celdas $Nreg_i$ viene dado por:

$$Nreg_i \sim Poisson(\lambda_i * S_i)$$

al ser registros oportunistas, es probable que no encontremos un número de registros suficientemente alto en cada celda como para poder ajustar una distribución de Poisson. Por ese motivo, utilizaremos este otro modelo:

$$y_i \sim Bernoulli(\psi_i) \quad \text{(Eq 4)}$$

$$cloglog(\psi_i) = log(\lambda_i) * log(S_i) \quad \text{(Eq 5)}$$

$$log(\lambda_i) = \beta_1 * alt_i + \beta_2 * bosque_i$$

donde y_i es un vector de longitud igual al número de celdas en nuestro área de estudio en el que pondremos un 0 si no se ha encontrado un registro, y un 1 cuando se haya encontrado al menos un registro. En nuestro juego de datos, contamos tan solo con 223 registros oportunistas (`nrow(registros[registros$det == 1,])`). Teniendo en cuenta que nuestro área de estudio es de 10000 celdas (`ncell(alt)`), los registros oportunistas ocupan apenas el 2% de nuestro territorio, por lo que se pueden considerar sucesos muy raros. Para este tipo de sucesos, la función vínculo *complementaria log-log* (*cloglog*) puede resultar útil: se trata de una función similar a *logit* que vincula una variable binaria con una continua, pero de forma asimétrica, aproximándose a cero infinitamente más despacio que cualquier otra función (*logit*, *probit*, etc.)[1]. En este modelo, S_i será la "probabilidad de encontrar un registro" parámetro que, en este caso, representa la mezcla de dos procesos: detectabilidad imperfecta y sesgo de muestreo. Este parámetro S_i es a su vez modelizado mediante:

$$logit(S_i) = \alpha_0 + \alpha_1 * dcarret_i \quad \text{(Eq 6)}$$

Es importante no confundir el *log* que acompaña a S_i en la Eq 5, que proviene del vínculo *cloglog*, del *logit* que acompaña a S_i en la Eq 6, el cual representa que S_i es una probabilidad y que por tanto debemos acotar entre 0 y 1.

Por último, conviene destacar que para que los parámetros de este modelo sean identificables, tanto λ_i como S_i deben tener al menos un predictor continuo "privado", esto es, al menos una covariable continua cada uno que no sea compartida entre ambos procesos. Para más información sobre este modelo se puede consultar **este post** de Mason Fidino (en inglés).

```
constants <- list(ncell = ncell(alt), nPO = nrow(registros))

data <- list(y = registros$det, alt = alt[], bosque = bosque[], dcarret = dcarret[])

# Definimos nuestro modelo
registros_model <- nimbleCode({

    # PRIORS
    b_alt ~ dnorm(0, 10)
```

[1]Para una recordar qué son y la que forma tienen las funciones vínculo **logit**, **probit** y **cloglog** se puede consultar este **enlace** (en inglés).

```
    b_bosque ~ dnorm(0, 10)
    a_intercept ~ dnorm(0, 10)
    a_dcarret ~ dnorm(0, 10)

    # LIKELIHOOD Variable de ESTADO o LATENTE: ABUNDANCIA
    for (i in 1:ncell) {
      # (Eq 1)
        n[i] ~ dpois(lambda[i])
      # (Eq 2)
        log(lambda[i]) <- b_alt * alt[i] + b_bosque * bosque[i]
    }

    # REGISTROS OPORTUNISTAS
    # Para cada una de las celdas de nuestro área de estudio...
    for (j in 1:ncell) {
      # que haya o no un registro de presencia sigue Bernoulli... (Eq 4)
        y[j] ~ dbern(psi[j])
      # con parámetro psi que depende... (Eq 5)
        cloglog(psi[j]) <- log(lambda[j]) + log(s[j])
      # de la distancia a la carretera mas cercana (Eq 6)
        logit(s[j]) <- a_intercept + a_dcarret * dcarret[j]
    }
})

# Valores de inicio
inits <- function() {
    list(n = rep(1, constants$nPO), b_alt = runif(1, -1, 1), b_bosque = runif(1,
        -1, 1), a_intercept = runif(1, -1, 1), a_dcarret = runif(1, -1,
        1))
}

monitores <- c("b_alt", "b_bosque", "a_intercept", "a_dcarret")

nc <- 1
nb <- 5000
ni <- nb + 50000

# MCMC
model <- nimbleModel(code = registros_model, data = data, constants = constants,
    inits = inits(), calculate = FALSE)
c_model <- compileNimble(model)
model_conf <- configureMCMC(model, useConjugacy = FALSE)
model_conf$addMonitors(monitores)
model_mcmc <- buildMCMC(model_conf)
c_model_mcmc <- compileNimble(model_mcmc, project = model)
samples_registros <- runMCMC(c_model_mcmc, nburnin = nb, niter = ni, nchains = nc)

# Predicciones y valores del MCMC
pred_Registros <- exp(mean(samples_registros[, 3]) * alt + mean(samples_registros[,
    4]) * bosque)
samples_Reg <- coda::mcmc(samples_registros)
```

Modelo integrado

Finalmente ajustaremos el modelo integrando ambas bases de datos. La clave de la aproximación de *joint-likelihood* es que ambos procesos observacionales (conteos repetidos y registros oportunistas) comparten el mismo proceso de estado o latente (la abundancia). Esto hace que compartan parámetros con él, por lo que integran o comparten información en un único modelo.

```
# Definimos nuestras constantes
constants <- list(ncell = ncell(alt), ncont = nrow(conteos), nsurveys = rep(4,
    nrow(conteos)), nPO = nrow(registros), cellCont = conteos$site_ID,
    cellReg = registros$ID)

# Definimos nuestros datos
data <- list(yConteos = as.matrix(conteos[, 2:5]), yRegistros = registros$det,
    alt = alt[], bosque = bosque[], dcarret = dcarret[])

# Definimos nuestro modelo
INT_model <- nimbleCode({

    # PRIORS
    b_alt ~ dnorm(0, 10)
    b_bosque ~ dnorm(0, 10)
    a_intercept ~ dnorm(0, 10)
    a_dcarret ~ dnorm(0, 10)
    p_binom ~ dbeta(1, 1)

    # LIKELIHOOD Variable de ESTADO o LATENTE: ABUNDANCIA (proceso compartido)
    for (i in 1:ncell) {
      # (Eq 1)
        n[i] ~ dpois(lambda[i])
      # (Eq 2)
        log(lambda[i]) <- b_alt * alt[i] + b_bosque * bosque[i]

    }

    # Proceso OBSERVACIONAL para CONTEOS REPETIDOS
    for (j in 1:ncont) {
        for (k in 1:nsurveys[j]) {
          # (Eq 3)
            yConteos[j, k] ~ dbin(p_binom, n[cellCont[j]])
        }
    }
    # Proceso OBSERVACIONAL para REGISTROS OPORTUNISTAS
    for (j in 1:ncell) {
      # (Eq 4)
        yRegistros[j] ~ dbern(psi[j])
      # (Eq 5)
        cloglog(psi[j]) <- log(lambda[j]) + log(s[j])
      # (Eq 6)
        logit(s[j]) <- a_intercept + a_dcarret * dcarret[j]
    }

})

# Valores de inicio
```

```
inits <- function() {
    base::list(n = rep(1, constants$ncell), b_alt = runif(1, -1, 1), b_bosque = runif(1,
        -1, 1), a_intercept = runif(1, -1, 1), a_dcarret = runif(1, -1,
        1), p_binom = runif(1, 0, 1))
}

# Monitores
monitores <- c("b_alt", "b_bosque", "a_intercept", "a_dcarret", "p_binom")

# MCMC
nc <- 1
nb <- 5000
ni <- nb + 50000

model <- nimbleModel(code = INT_model, data = data, constants = constants,
    inits = inits(), calculate = FALSE)
c_model <- compileNimble(model)
model_conf <- configureMCMC(model, useConjugacy = FALSE)
model_conf$addMonitors(monitores)
model_mcmc <- buildMCMC(model_conf)
c_model_mcmc <- compileNimble(model_mcmc, project = model)
samples_INT <- runMCMC(c_model_mcmc, nburnin = nb, niter = ni, nchains = nc)

# Predicciones y valores del MCMC
pred_INT <- exp(mean(samples_INT[, 3]) * alt + mean(samples_INT[, 4]) *
    bosque)
samples_Int <- coda::mcmc(samples_INT)
```

Comparativa de modelos

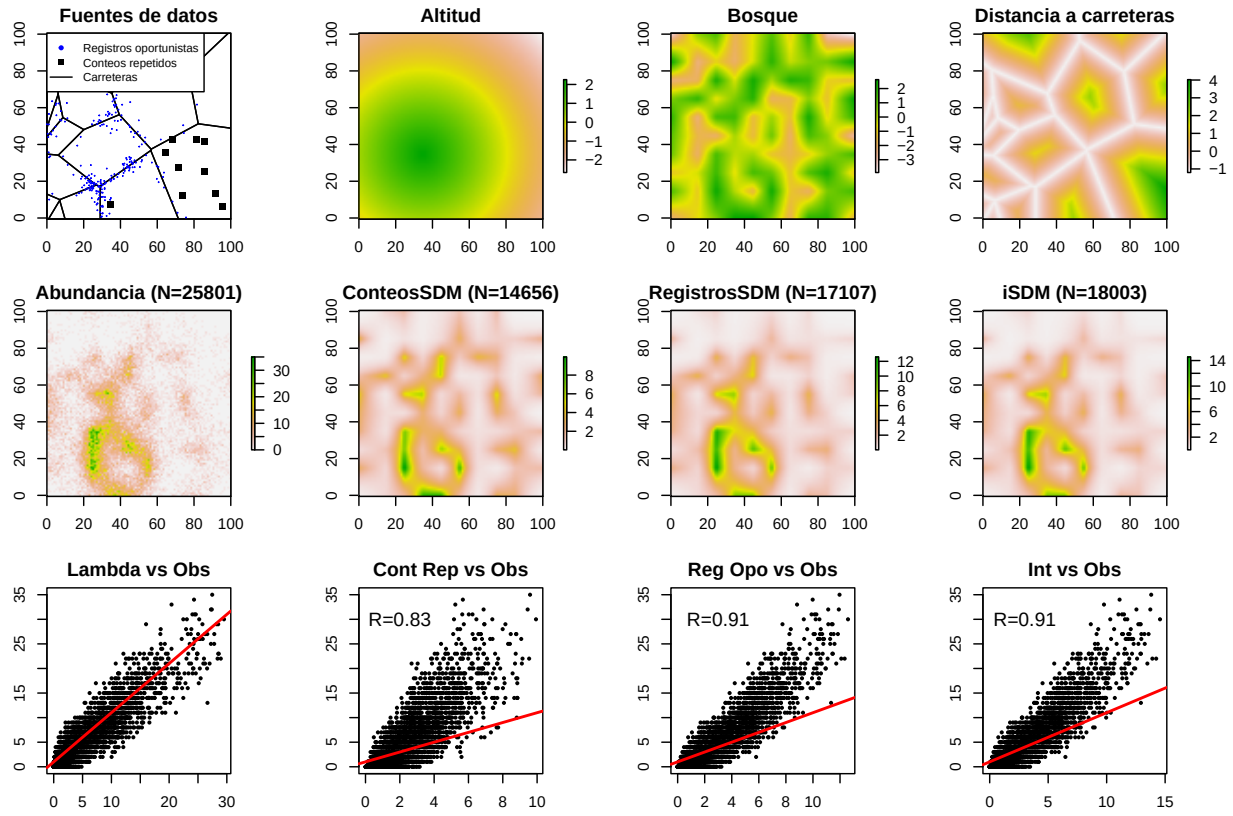
A la hora de comparar los modelos existen diferentes formas de hacerlo. En este caso nos fijaremos en la predicción espacial y en los valores de los coeficientes obtenidos por cada uno de los modelos. Una simulación debe entenderse como un experimento controlado en un laboratorio. La ventaja de aprender con ayuda de simulaciones es que conocemos exactamente los valores reales de cada parámetro (ya que los hemos decidido nosotros), así como la distribución real de la abundancia de nuestros individuos, por lo que podremos valorar el rendimiento de cada una de nuestras aproximaciones.

Predicciones espaciales

A forma de resumen, representaremos gráficamente los resultados más importantes relacionados con las predicciones espaciales:

```
par(mfrow = c(3, 4), mar = c(3, 3, 2, 4))
plot(abun, alpha = 0, legend = F, main = "Fuentes de datos")
lines(carret)
points(ind[ind$det == 1, 1:2], col = "blue", cex = 0.1, pch = 20)
points(xyFromCell(abun, site_ID), pch = 15, cex = 1)
legend("topleft", legend = c("Registros oportunistas", "Conteos repetidos",
    "Carreteras"), pch = c(20, 15, NA), lty = c(NA, NA, 1), col = c("blue",
    "black", "black"), bg="white", cex = .8)
plot(alt, main = "Altitud")
plot(bosque, main = "Bosque")
plot(dcarret, main = "Distancia a carreteras")
```

```
plot(abun, main = paste("Abundancia (N=", round(sum(abun[]), 0), ")", sep = ""))
plot(pred_Conteos, main = paste("ConteosSDM (N=", round(sum(pred_Conteos[]),
    0), ")", sep = ""))
plot(pred_Registros, main = paste("RegistrosSDM (N=", round(sum(pred_Registros[]),
    0), ")", sep = ""))
plot(pred_INT, main = paste("iSDM (N=", round(sum(pred_INT[]), 0), ")", sep = ""))
plot(rZ[], abun[], ylab = "Observados", xlab = "Predichos", main = "Lambda vs Obs",
    pch = 16, cex = 0.6)
abline(a = 1, b = 1, col = "red", lwd = 2)
plot(pred_Conteos[], abun[], ylab = "Observados", xlab = "Predichos",
     main = "Cont Rep vs Obs", pch = 16, cex = 0.6)
text(2, 30, paste("R=", round(cor(abun[], pred_Conteos[]), 2), sep = ""),
    cex = 1.2)
abline(a = 1, b = 1, col = "red", lwd = 2)
plot(pred_Registros[], abun[], ylab = "Observados", xlab = "Predichos",
    main = "Reg Opo vs Obs", pch = 16, cex = 0.6)
text(3, 30, paste("R=", round(cor(abun[], pred_Registros[]), 2), sep = ""),
    cex = 1.2)
abline(a = 1, b = 1, col = "red", lwd = 2)
plot(pred_INT[], abun[], ylab = "Observados", xlab = "Predichos", main = "Int vs Obs",
    pch = 16, cex = 0.6)
text(3, 30, paste("R=", round(cor(abun[], pred_INT[]), 2), sep = ""), cex = 1.2)
abline(a = 1, b = 1, col = "red", lwd = 2)
```



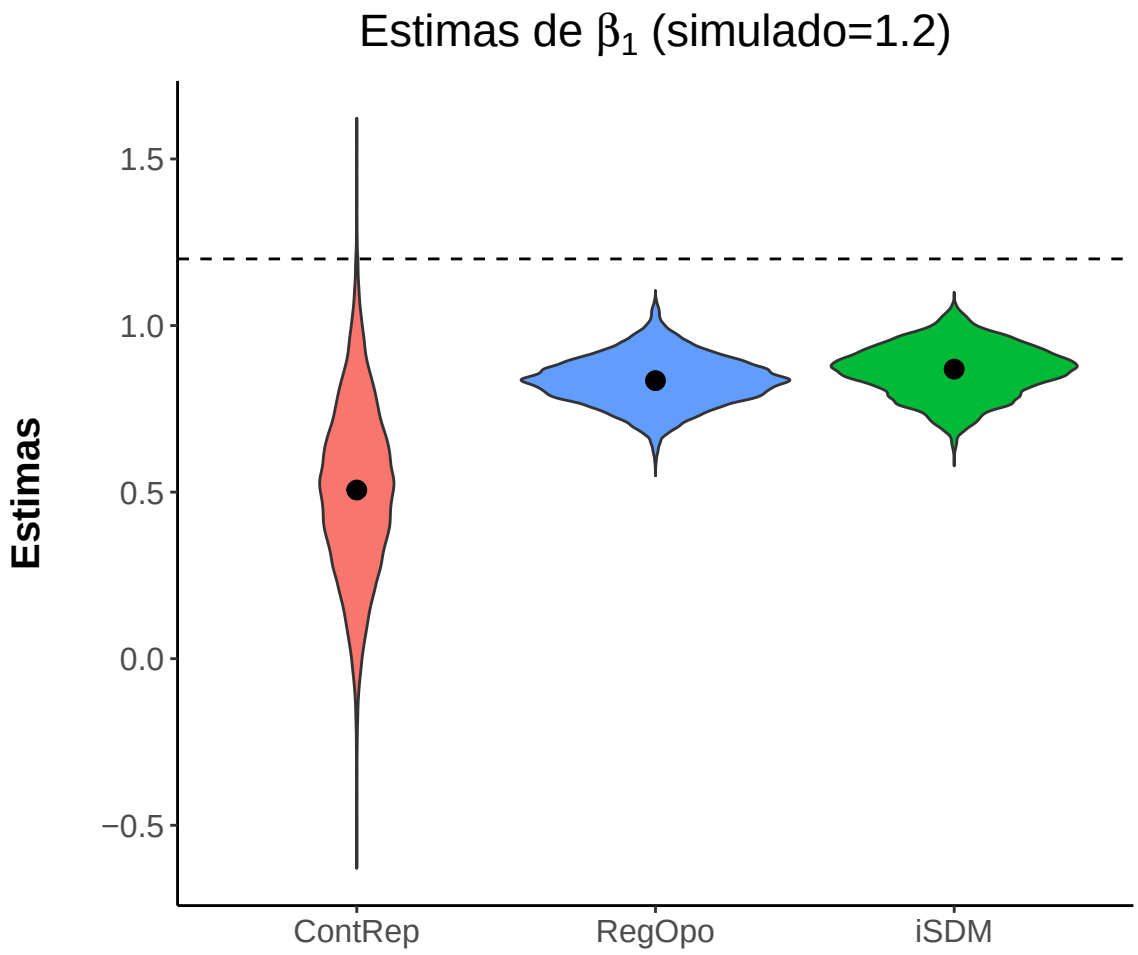
Como podemos ver en la segunda fila, el patrón espacial de la abundancia predicha por cada uno de los modelos fue, en general, muy similar. Sin embargo, en cuanto a la abundancia total estimada, el modelo

integrado (iSDM), con una predicción de 18,003 individuos fue el que más se acercó al número real de animales simulados (25,801). Tanto el modelo a partir de registros oportunistas como el integrado obtuvieron una muy alta correlación con la abundancia simulada (R = 0.91), aunque cabe destacar que el modelo con conteos repetidos, con tan solo 10 sitios muestreados, también obtuvo una alta correlación con la abundancia (R = 0.83).

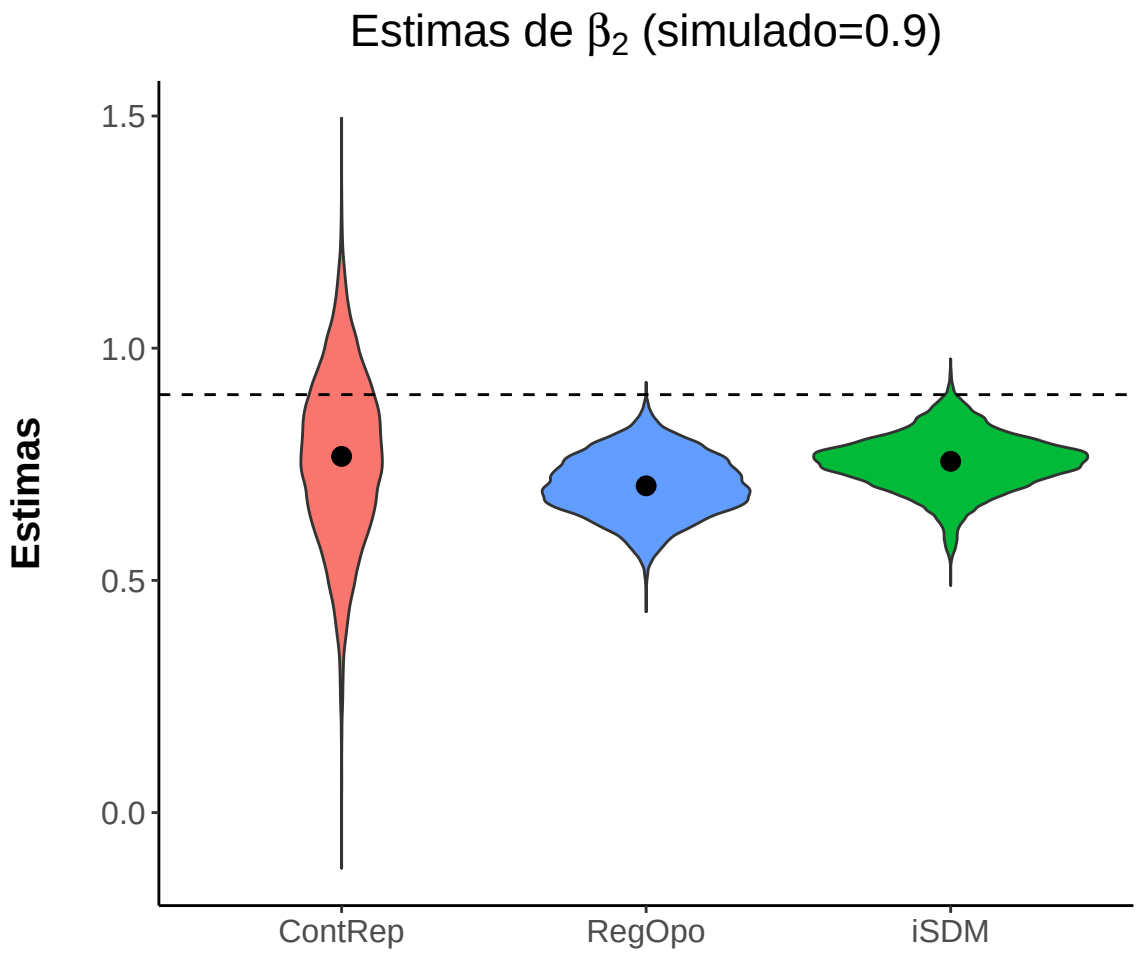
Coeficientes (*estimates*)

A continuación se muestran las distribuciones a posteriori de los coeficientes β_1 y β_2 obtenidas a partir de los MCMC. La linea horizontal punteada marca el valor que fue simulado. Lo primero que podemos observar es que la precisión de las estimas de modelo usando solo los conteos repetidos es mucho menor que la de los otros modelos. Esto es comprensible, dado el pequeño tamaño muestral de ese juego de datos. Además, podemos apreciar que las estimas del modelo integrado son ligeramente mejores que las del modelo utilizando tan solo los registros oportunistas. Estas ligeras mejoras producen una mejor predicción del tamaño poblacional total en nuestro área de estudio, como hemos visto en la sección anterior.

```
alt_Est <- data.frame(estimates = c(samples_Cont[, 1], samples_Reg[, 3], samples_Int[,
    3]), group = rep(c("ContRep", "RegOpo", "iSDM"), 1, each = 50000))
bos_Est <- data.frame(estimates = c(samples_Cont[, 2], samples_Reg[, 4], samples_Int[,
    4]), group = rep(c("ContRep", "RegOpo", "iSDM"), 1, each = 50000))

a_Est <- ggplot(alt_Est, aes(x = factor(group, levels = c("ContRep", "RegOpo",
    "iSDM")), y = estimates, fill = group)) + geom_violin(trim = F) +
  labs(title = expression(paste("Estimas de ",
    beta[1], " (simulado=1.2)")), x = "", y = "Estimas") + stat_summary(fun = median,
    geom = "point", size = 3, color = "black")

print(a_Est + theme_classic() + geom_hline(yintercept = 1.2, linetype = "dashed") +
    theme(axis.text = element_text(size = 12), axis.title = element_text(size = 15,
        face = "bold", margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.y = element_text(margin = margin(t = 0,
        r = 20, b = 0, l = 0)), axis.title.x = element_text(margin = margin(t = 20,
        r = 0, b = 0, l = 0)), legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 17, face = "bold")))
```




```
b_Est <- ggplot(bos_Est, aes(x = factor(group, levels = c("ContRep", "RegOpo",
    "iSDM")), y = estimates, fill = group)) + geom_violin(trim = F) +
  labs(title = expression(paste("Estimas de ",
    beta[2], " (simulado=0.9)")), x = "", y = "Estimas") + stat_summary(fun = median,
    geom = "point", size = 3, color = "black")

print(b_Est + theme_classic() + geom_hline(yintercept = 0.9, linetype = "dashed") +
    theme(axis.text = element_text(size = 12), axis.title = element_text(size = 15,
        face = "bold", margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.y = element_text(margin = margin(t = 0,
        r = 20, b = 0, l = 0)), axis.title.x = element_text(margin = margin(t = 20,
        r = 0, b = 0, l = 0)), legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 17, face = "bold")))
```



Referencias

Dorazio, Robert M. 2014. "Accounting for Imperfect Detection and Survey Bias in Statistical Analysis of Presence-Only Data." *Global Ecology and Biogeography* 23 (12): 1472–84.

Fithian, William, Jane Elith, Trevor Hastie, and David A Keith. 2015. "Bias Correction in Species Distribution Models: Pooling Survey and Collection Data for Multiple Species." *Methods in Ecology and Evolution* 6 (4): 424–38.

Royle, J Andrew. 2004. "N-Mixture Models for Estimating Population Size from Spatially Replicated Counts." *Biometrics* 60 (1): 108–15.

Valpine, Perry de, Daniel Turek, Christopher J Paciorek, Clifford Anderson-Bergman, Duncan Temple Lang, and Rastislav Bodik. 2017. "Programming with Models: Writing Statistical Algorithms for General Model Structures with NIMBLE." *Journal of Computational and Graphical Statistics* 26 (2): 403–13.