Simulación y análisis de ocupación

Entendiendo las simulaciones y el modelo básico de ocupación

Diego J. Lizcano, Ph.D.

2020-04-20

Índice general

Capítulo 1

Prerequisitos

Este es un *libro-tutorial* escrito con **Markdown**

Desde R y R studio, usando los paquetes 'bookdown', 'knitr'y 'rmarkdown'.

Este *libro-tutorial* hace parte del mini curso de métodos de ocupación con R.

Antes de comenzar por favor instale el programa JAGS en su computadora, posteriormente desde R studio instale los paquetes unmarked, raster, spatstat, jagsUI, mcmcplots y ggmcmc

```
install.packages("unmarked", dependencies = TRUE)
install.packages("raster", "spatstat", "jagsUI", "mcmcplots", "ggmcmc", dependencies =
```


Capítulo 2

Introducción

En este libro-tutorial realizaremos simulaciones de datos para modelos de ocupación, bajo el modelo estático de una sola estación, desarrollado originalmente por Daryl MacKenzie (**?**). El modelo básico de ocupación, es la piedra angular del muestreo y monitoreo biológico moderno!.

El objetivo de este documento es entender la versatilidad y el poder de las simulaciones con R en el capítulo **??**. Aprenderemos el principio biologico y el mecanismo bajo el cual funciona el modelo básico de ocupación en el capítulo **??**. Realizaremos un ejemplo sencillo en el capítulo **??** para un venado del Parque Nacional Machalilla. Posteriormente empacaremos el código que genera los datos de ocupación, en una sola función en el Capítulo **??**. Esta función nos permitirá simular datos bajo diferentes escenarios, los cuales analizaremos con los estimadores de máxima verosimilitud de la función occu del paquete unmarked en el Capítulo **??**. Posteriormente analizaremos los mismos datos bajo estimadores bayesianos en el Capítulo **??**, para al final comparar la precisión de ambas aproximaciones, máxima verosimilitud y bayesiana.

2.0.0.1 Créditos

Gran parte del concepto de este documento, el código y el texto han sido adaptados de los libros de Marc Kery (**???**).

Capítulo 3

Por qué simular?

3.1 por qué son útiles las simulaciones:

1. Al hacer simulaciones se conocen los parámetros verdaderos, así que podremos asegurarnos que el código que ejecutamos (R o BUGS) estima lo que queremos, y que los estimados son iguales o se acercan a los parámetros verdaderos, permitiendo depurar errores en el código.
2. Podemos calibrar un modelo derivado y/o más complejo más fácilmente. Las simulaciones pueden ser vistas como un experimento controlado, o como versiones simplificadas de un sistema real, en el cual podemos probar como varían ciertos parámetros que afectan los estimados de otros parámetros. Realizar experimentos controlados en el mundo real es muchas veces impráctico o imposible en ecología, así que la simulación es la forma más coherente de estudiar el sistema ecológico.
3. Se experimenta de primera mano el error de muestreo y se convierte en un fantástico proceso de aprendizaje.
4. Podemos verificar la calidad (frecuentista) de los estimados, así como la precisión y el efecto del tamaño muestreal, computando la diferencia entre la media del estimado y el valor real (sesgo) y la varianza del estimado (la precisión).
5. Es la forma más flexible y directa de realizar un análisis de poder, resolviendo el gran problema de determinar el tamaño de la muestra necesario para detectar un efecto de cierta magnitud, con una probabilidad dada.

6. Podemos visualizar que tan identificables son los parámetros en modelos más complejos.
7. Podemos verificar que tan robusto es el modelo a violaciones de lo que se asume.
8. Al ser capaces de simular datos bajo cierto modelo, se garantiza que uno

entiende el modelo, sus restricciones y limitaciones.

Capítulo 4

La ocupación de hábitat

Obtener datos para estudios de poblaciones animales, es costoso y dispendioso, y no siempre se puede medir la densidad poblacional o parámetros demográficos como natalidad o mortalidad. Es por eso que la estimación de ocupación de hábitat (ψ) es una buena herramienta de estudio, ya que es un reflejo de otros parámetros poblacionales importantes como la abundancia y densidad, que requieren de un elevado número de registros, con los costos económicos y logísticos que conlleva. Adicionalmente y debido a que la detectabilidad (p) en animales silvestres no es completa, el uso de los datos crudos genera subestimaciones de la ocupación de hábitat. Con el empleo de muestreos repetidos, es posible generar estimaciones de detectabilidad y, con esta estimación, obtener valores no sesgados de la ocupación del hábitat. Los métodos de análisis de la ocupación fueron inicialmente desarrollados por (**?**) y posteriormente expandidos por otros autores (**??????**). Este tipo de modelos permiten realizar inferencias acerca de los efectos de variables continuas y categóricas sobre la ocupación de hábitat. Además, sí los muestreos se realizan a través de períodos largos de tiempo, también es posible estimar tasas de extinción y recolonización, que son útiles en estudios de metapoblaciones (**?**). Este es un campo de gran desarrollo en bioestadística que ha producido una gran explosión de estudios que usan la ocupación teniendo en cuenta la detectabilidad (**??????**).

Capítulo 5

Nuestro ejemplo:

El set de datos que vamos a simular, imita la forma espacial y temporal como imaginamos se originan las medidas repetidas de presencia ausencia en ecología. Las cuales son una combinación de un proceso ecológico y un proceso de observación. El primer proceso contiene los mecanismos bajo los cuales se originan patrones espacio-temporales de distribución, mientras que el segundo proceso contiene las diferentes facetas en las cuales se originan fuentes de error al tomar los datos.

Para ser más concretos vamos a llamar a nuestra especie imaginaria con un nombre real. La llamaremos el venado de cola blanca (*Odocoileus virginianus*), un mamífero grande y común, ampliamente distribuido en el continente Americano, y de preocupación menor en términos de conservación.

El Venado de cola blanca (*Odocoileus virginianus*) nuestra especie de

interes para este ejemplo. Foto del proyecto fauna de Manabi http://faunamanabi.github.io

El set de datos contiene J datos replicados de detección o no detección de la especie en M sitios, teniendo en cuenta que asumimos que es una población cerrada ('closure' assumption). Es decir que durante el muestreo no ocurrieron cambios por nacimientos, muertes inmigración o emigración. En otras palabras, el muestreo fue corto en tiempo y la ocurrencia de la especie z no cambió por efectos demográficos.

Claramente debemos distinguir dos procesos, el primero es el proceso ecológico, el cual genera (parcialmente) un estado latente de la ocurrencia z. El segundo es el proceso de observación, el cual produce los datos observados de detección o no detección del venado. Aquí asumimos que el proceso de observación está gobernado por un mecanismo de detección imperfecta. Es decir algunos venados pudieron haberse escapado a mi observación, lo cual genera falsos negativos. También asumimos que los falsos positivos están ausentes, es decir que todo lo que identifico como venado, es efectivamente un venado. Para hacer más realista el ejemplo incluimos los efectos de la altitud y la cobertura de bosque en la ocurrencia, como factores que afectan la ocurrencia linealmente, disminuyéndola para el caso de la elevación y que incrementan la ocupación linealmente para el caso de la cobertura de bosques. Al final las dos variables interactúan negativamente entre sí. Esos efectos se introducen en la ocurrencia en escala logarítmica como tradicionalmente se hace un modelo lineal generalizado (GLM).

En nuestra simulación vamos a hacer explícito que no es posible detectar a la totalidad de los venados de un sitio de muestreo, así que estamos enfrentando un tipo de error que nos hace sub-estimar la abundancia de la población. Hay muchas razones por las cuales fallamos en detectar un individuo en la naturaleza, porque nos distrajimos mientras el venado pasó, porque los binoculares no tenían el aumento suficiente, o simplemente porque el venado se escondió detrás de un árbol al sentir nuestro olor, o por alguna otra razón. De esta forma nosotros vamos a registrar la presencia (z=1) con una probabilidad de detección p la cual también vamos a hacerla dependiente (en la escala logarítmica) de la altitud y de una co-variable que afecta la detección, la temperatura. En términos generales los animales son más difíciles de observar cuando la temperatura es más alta y por lo general entre más altitud la temperatura disminuye. De esta forma asumimos que la detección está relacionada negativamente con la altitud y la temperatura. Pero también hay que tener en cuenta que el efecto negativo en p también puede ser mediado por una disminución en la abundancia con la altitud, el cual también causa que la probabilidad de ocupación disminuya con la altitud. Tenga en cuenta que una co-variable, la altitud afecta a ambos procesos, el ecológico (la ocurrencia) y al proceso de observación (la probabilidad de detección). Esto tiene un propósito, y es probable que pase en la naturaleza muchas veces. Los modelos de ocupación tienen una base "mecanicista" produciendo variación espacial en la abundancia. Es decir tendremos sitios con mayor abundancia y otros con menor abundancia. Pero los modelos jerárquicos cómo el

que estamos por construir, son capaces de desentrañar esas relaciones complejas entre ocurrencia y probabilidad de detección (**???**). Finalmente para este primer ejemplo vamos a dejar por fuera el efecto de la interacción entre la altitud y la temperatura, ajustándolo a cero. Luego podremos variar este parámetro para considerar ese efecto. En resumen vamos a generar datos bajo el siguiente modelo, donde los sitios son indexados cómo i y los conteos repetidos en el sitio van a ser referidos cómo j.

5.0.1 Modelo Ecológico:

$$z_i = Bernoulli(\psi_i) \tag{5.1}$$

$$logit(\psi_i) = \beta_0 + \beta_1 * Altitud_i + \beta_2 * CovBosque_i + \beta_3 * Altitud_i * CovBosque_i \tag{5.2}$$

5.0.2 Modelo de Observación:

$$y_{ij} = Bernoulli(z_i * p_{ij}) \tag{5.3}$$

$$logit(p_{ij}) = \alpha_0 + \alpha_1 * Altitud_i + \alpha_2 * Temperatura_{ij} + \alpha_3 * Altitud_i * Temperatura_{ij} \tag{5.4}$$

Donde ψ es la ocupación y p la probabilidad de detección. Con β como el coeficiente de la regresión para las co-variables de la ocupación y α el coeficiente de regresión para las co-variables de la detección.

Vamos a generar datos desde "dentro hacia afuera" y desde arriba hacia abajo. Para esto, primero escogemos el tamaño de la muestra y creamos los valores para las co-variables. Segundo, seleccionamos los valores de los parámetros del modelo ecológico (la ocupación) y ensamblamos la ocurrencia esperada (el parámetro ψ, ocupación) y posteriormente obtenemos la variable al azar z la cual tiene distribución Bernoulli. Tercero, seleccionamos los valores de los parámetros del modelo de observación (la detección), para ensamblar la probabilidad de detección p y obtener el segundo set de una variable al azar y (detección observada o no observada de un venado) la cual también tiene distribución Bernoulli.

Para simular los datos usaremos el lenguaje de programación estadística R (**?**), el cual provee una gran variedad de técnicas gráficas y estadísticas de modelación y un gran ecosistema de paquetes para análisis estadístico y ecológico. Si aún no lo ha hecho, baje e instale R en su computadora, posteriormente haga lo mismo con RStudio.

5.1 Pasos iniciales: tamaño de la muestra y valores de las co-variables

Inicie RStudio, copie, pegue y ejecute los comandos de la ventana gris.

Primero escogemos el tamaño de la muestra, el número de sitios y el número de medidas repetidas (número de visitas) de presencia/ausencia en cada sitio.

```
M <- 60  # Número de réplicas espaciales (sitios)
J <- 30  # Número de réplicas temporales (conteos repetidos)
```

Luego creamos los valores para las co-variables. Tenemos altitud y cobertura de bosque como co-variables de cada sitio. Ellas difieren de sitio a sitio pero para cada muestreo son las mismas. Mientras que la temperatura es una co-variable de la observación, así que si varía en cada muestreo y también en cada sitio. Recuerde que el sub índice i se refiere al sitio y el j a cada muestreo. Para simplificar las cosas nuestras co-variables van a tener una distribución normal con una media centrada en cero y no se van a extender muy lejos en cada lado del cero. En análisis de datos reales tendremos que estandarizar las co-variables para evitar problemas numéricos de diferencia en las escalas de las co-variables y poder calcular el valor de máxima verosimilitud (ML), así como también para obtener convergencia en las cadenas de Markov del modelo Bayesiano. Aquí vamos a ignorar un hecho de la vida real, y es que las co-variables no son totalmente independientes la una de la otra, es decir en la naturaleza la cobertura boscosa puede estar relacionada con la altitud, pero esto no va a ser relevante, por ahora.

Para inicializar el generador de números aleatorios y obtener siempre los mismos resultados podemos adicionar la siguiente línea:

```
set.seed(24) # Can choose seed of your choice
```

De esta forma podremos obtener siempre los mismos estimados. Pero luego cuando queramos obtener el error de muestreo deberemos remover esa línea. Para este ejemplo generaremos valores para las covariables centrados en cero y variando de -1 a 1.

```
elev <- runif(n = M, -1, 1)              # Scaled elevation of a site
forest <- runif(n = M, -1, 1)            # Scaled forest cover at each site
temp <- array(runif(n = M*J, -1, 1), dim = c(M, J)) # Scaled temperature
```

5.2 Simulando el proceso ecológico y su resultado: la ocurrencia del venado

Para simular la ocurrencia del venado en cada sitio, escogemos los valores para los parámetros que gobiernan la variación espacial en la ocurrencia β_0 a β_3. El primer parámetro es la ocurrencia promedio esperada del venado (probabilidad de ocupación) cuando todas las co-variables tienen un valor de cero, en otras palabras el intercepto del modelo de ocurrencia. Preferimos pensar en los venados en términos de su ocurrencia en lugar de logit(ocurrencia). Aquí nosotros escogemos el intercepto de la ocupación primero y luego lo transformamos de la escala logarítmica con la función de enlace logit.

```
mean.occupancy <- 0.60          # Mean expected occurrence of deer
beta0 <- plogis(mean.occupancy) # Same on logit scale (= logit-scale intercept)
beta1 <- -2                     # Effect (slope) of elevation
beta2 <- 2                      # Effect (slope) of forest cover
beta3 <- 1                      # Interaction effect (slope) of elev and forest
```

Aquí aplicamos el modelo lineal (a la escala logarítmica) y obtenemos la transformación logit de la probabilidad de ocupación, la cual invertimos con la transformación logit para obtener la ocupación del venado y graficar todo.

```
logit.psi <- beta0 + beta1 * elev + beta2 * forest + beta3 * elev * forest
psi <- plogis(logit.psi)      # Inverse link transformation

# par()               # view current settings
opar <- par()       # make a copy of current settings
par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
curve(plogis(beta0 + beta1*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1),
      xlab = "Altitud", ylab = "psi", lwd = 2)
text(0.9, 0.95, "A", cex = 1.5)
plot(elev, psi, frame.plot = FALSE, ylim = c(0, 1), xlab = "Altitud", ylab = "")
text(0.9, 0.95, "B", cex = 1.5)
curve(plogis(beta0 + beta2*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1),
      xlab = "Forest cover", ylab = "psi", lwd = 2)
text(-0.9, 0.95, "C", cex = 1.5)
plot(forest, psi, frame.plot = FALSE, ylim = c(0, 1), xlab = "Forest cover", ylab = "")
text(-0.9, 0.95, "D", cex = 1.5)
```

Figura 5.1: Dos formas de mostrar la relación entre la probabilidad de ocurrencia de los venados y las co-variables. (A) Relación entre psi y altitud para un valor constante (media igual a cero) de cobertura boscosa. (B) Relación entre psi y la altitud en un valor observado de cobertura boscosa. (C) relación psi cobertura boscosa para una altitud constante (en la media de cero). (D) Relación psi cobertura boscosa para el valor observado de altitud.

```
# dev.off()
par(opar)          # restore original par settings
```

Para mostrar mejor la relación conjunta entre las dos covariables y psi, debemos realizar un diagrama de superficie. Aquí no hemos cambiado nada de la simulación, solo le hemos agregado más datos para visualizar mejor.

```
# Compute expected occurrence for a grid of elevation and forest cover
cov1 <- seq(-1, 1, , 100)                       # Values for elevation
cov2 <- seq(-1, 1, , 100)                       # Values for forest cover
psi.matrix <- array(NA, dim = c(100, 100))      # Prediction matrix, for every
# combination of values of elevation and forest cover

for(i in 1:100){
   for(j in 1:100){
      psi.matrix[i, j] <- plogis(beta0 +
                                 beta1 * cov1[i] +
                                 beta2 * cov2[j] +
                                 beta3 * cov1[i] * cov2[j] )
   }
}

mapPalette <- colorRampPalette(c("grey", "yellow", "orange", "red"))
image(x = cov1, y = cov2, z = psi.matrix, col = mapPalette(100), xlab = "Altitud",
      ylab = "Forest cover", cex.lab = 1.2)
contour(x = cov1, y = cov2, z = psi.matrix, add = TRUE, lwd = 1)
matpoints(elev, forest, pch="+", cex=0.8)
```

Figura 5.2: Relación construida ente los datos simulados de la ocurrencia esperada (ocupación) del venado (psi) representada con la escala de color de gris a rojo, contra la altitud y la cobertura boscosa simultáneamente. En este caso la interacción entre las dos covariables está dada por el valor de beta3 = 1 que hemos establecido anteriormente.

Hasta ahora no hemos introducido ninguna variación estocástica en la relación entre la ocurrencia del venado y las covariables. Para hacer esto debemos hacer uso de algunos modelos estadísticos o distribuciones estadísticas, para describir la variabilidad al azar alrededor del valor esperado de psi. La forma típica de introducir esta variación al azar es obtener la ocurrencia de venados en cada sitio i, z_i, de una distribución Bernoulli con los valores esperados (ψ_i).

5.2.1 Por qué Bernoulli?

En el proceso ecológico z_i la ocurrencia del venado esta representado por una distribución de tipo Bernoulli donde el venado esté presente en un sitio representado como la ocupación ψ en un sitio en que esta presente, o no esta presente 1-ψ. La distribución Bernoulli es un caso especial de la distribución binomial, y su mejor ejemplo es el lanzamiento de una moneda una sola vez. Si requiere una explicación más extensa, básica, detallada y con más ejemplos le recomiendo visitar khanacademy.

```
z <- rbinom(n = M, size = 1, prob = psi)  # Realised occurrence. A Bernoulli
sum(z)                                    # Total number of occupied sites
```

```
## [1] 40
```

```
table(z)                                  # Frequency distribution of deer occurrence
```

```
## z
##  0  1
## 20 40
```

Aquí hemos creado el resultado del proceso ecológico: ocurrencia específica para cada sitio z_1. Vemos que 20 sitios no están ocupados y que los restantes 40 si están ocupados.

5.3 Simulando el proceso de observación y su resultado: la detección

La ocurrencia z no es lo que normalmente vemos, ya que hay un chance de que fallemos en observar un individuo. De ahí que haya una medida binaria de error cuando medimos la ocurrencia (lo observamos o no lo observamos). Nosotros asumimos que podemos hacer únicamente una de las dos posibles observaciones (si, no), pero pudimos haber perdido la observación de un venado en algún sitio, entonces la probabilidad de detección es menor que uno y la medida de error es afectada por la cobertura de bosque y la temperatura. Hay que tener en cuenta que nunca vamos a registrar la presencia de un venado cuando en realidad no hay venados. En otras palabras estamos asumiendo que no tenemos falsos positivos. Para hacer explícito que tenemos un efecto de interacción entre dos co-variables en nuestros datos, vamos a permitir un efecto de la interacción en el código, pero ajustado a cero y de esta forma sin efecto en el modelo que genera los datos. Primero seleccionamos los valores para α_0 hasta α_3, donde el primero es la probabilidad de detección para el venado, en la escala logit, cuando todas las co-variables de la detección tienen un valor de cero. Hemos escogido el intercepto del modelo de detección y luego lo transformamos con la función de enlace plogis. Esto no es lo mismo que la probabilidad de detección media, la

cual es más alta en nuestro modelo de simulación, como veremos más adelante.

```
mean.detection <- 0.3             # Mean expected detection
alpha0 <- qlogis(mean.detection) # same on logit scale (intercept)
alpha1 <- -1                      # Effect (slope) of elevation
alpha2 <- -3                      # Effect (slope) of temperature
alpha3 <- 0                       # Interaction effect (slope) of elevevation and tempe
```

Aplicando el modelo lineal, tenemos el logit de la probabilidad de detección del venado para cada sitio y muestreo, y aplicándole la transformación inversa (plogis), obtenemos una matriz de las dimensiones 60 por 30 con la probabilidad de detección para cada sitio i y muestreo j. Finalmente, graficamos las relaciones para la probabilidad de detección en los datos.

```
logit.p <- alpha0 + alpha1 * elev + alpha2 * temp + alpha3 * elev * temp
p <- plogis(logit.p)              # Inverse link transform
mean(p)                           # average per-site p is about 0.39
```

```
## [1] 0.3928068
```

```
par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
curve(plogis(alpha0 + alpha1*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1
      xlab = "Altitud", ylab = "p", lwd = 2)
text(-0.9, 1.05, "A", cex = 1.5)
matplot(elev, p, pch = "*", frame.plot = FALSE, ylim = c(0, 1.1), xlab = "Altitud",
        ylab = "")
text(-0.9, 1.05, "B", cex = 1.5)
curve(plogis(alpha0 + alpha2*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1
      xlab = "Temperature", ylab = "p", lwd = 2)
text(-0.9, 1.05, "C", cex = 1.5)
matplot(temp, p, pch = "*", frame.plot = FALSE, ylim = c(0, 1.1), xlab = "Temperature"
        ylab = "p")
text(-0.9, 1.05, "D", cex = 1.5)
```

\begin{figure} \caption[fig4]{Dos formas de mostrar las relaciones entre la probabilidad de detección esperada del venado (p) y las dos variables altitud y temperatura. (A) Relación p y altitud para temperatura constante (en el valor medio, que es igual a cero). (B) Relación entre p y la altitud en el valor observado de cantidad de temperatura. (C) Relación entre p y temperatura para un valor constante de altitud (en la altitud media igual a cero). (D) Relación entre p y temperatura para un valor observado de altitud.} \end{figure}

De forma similar vamos a producir una gráfica de una superficie con la relación conjunta entre la altitud, la temperatura y la probabilidad de detección del venado (p), actuando simultáneamente. La relación en la escala logarítmica es representada por un plano con pendiente que representa la interacción entre la elevación y la cobertura de bosque.

```
# Compute expected detection probability for a grid of elevation and temperature
cov1 <- seq(-1, 1,,100)                  # Values of elevation
cov2 <- seq(-1,1,,100)                   # Values of temperature
p.matrix <- array(NA, dim = c(100, 100)) # Prediction matrix which combines
# every value in cov 1 with every other in cov2
for(i in 1:100){
   for(j in 1:100){
      p.matrix[i, j] <- plogis(alpha0 + alpha1 * cov1[i] +
                               alpha2 * cov2[j] +
                               alpha3 * cov1[i] * cov2[j])
   }
}
image(x = cov1, y = cov2, z = p.matrix, col = mapPalette(100), xlab = "Altitud",
      ylab = "Temperature", cex.lab = 1.2)
contour(x = cov1, y = cov2, z = p.matrix, add = TRUE, lwd = 1)
matpoints(elev, temp, pch="+", cex=0.7, col = "black")
```

Figura 5.3: Relación construida ente los datos simulados de la probabilidad de detección esperada (detectabilidad) del venado (p) representada con la escala de color de gris a rojo, contra la altitud y la temperatura simultáneamente. En este caso la interacción entre las dos covariables tiene una relación lineal que esta dada por el valor de alpha3 = 0 que hemos establecido anteriormente.

Hasta acá hemos modelado los dos procesos el ecológico z y el de observación p por aparte. Ahora tendremos que ponerlos juntos, y para esto multiplicamos el resultado del proceso ecológico por la probabilidad de detección dentro de una distribución Bernoulli.

5.3.1 Uniendo los dos procesos el ecológico y el de observación

Cuando "medimos" la ocurrencia, la detección imperfecta, representa una fuente de error con una distribución de tipo Bernoulli (por ejemplo la presencia del venado en un sitio en que es detectado con una probabilidad p, o no es detectado como 1-p). Al aplicar este proceso de observación producimos medidas repetidas de la presencia o ausencia (1 o 0) del venado en cada sitio. Recuerde que la distribución Bernoulli es un caso especial de la distribución binomial, y su mejor ejemplo es el lanzamiento de una moneda una sola vez.

En este momento estamos estableciendo la jerarquía en el modelo jerárquico. Aca estamos anidando el proceso ecológico "dentro" del proceso de observación.

```
y <- matrix(NA, nrow = M, ncol = J)      # Prepare array for counts
for (i in 1:J){                          # Generate counts
```

```
   y[,i] <- rbinom(n = M, size = 1, prob = z*p[,i])   # this is the Bernoulli
}
```

Hasta acá hemos simulado la presencia/ausencia del venado de cola blanca en 60 sitios durante 30 sesiones de muestreo. Veamos que contienen las tablas. Recuerde que los sitios están en las filas y los muestreos repetidos en las columnas. Para comparar, mostraremos la verdadera ocurrencia en la primera columna para 30 sitios y solo cinco muestreos.

```
library(knitr)
kable(as.data.frame(head(cbind("True Presence/Absence" = z,
           "1st survey" = y[,1],
           "2nd survey" = y[,2],
           "3rd survey" = y[,3],
           "4th survey" = y[,4],
           "5th survey" = y[,5]), 30)) )   # First 30 rows (= sites)
```

True Presence/Absence	1st survey	2nd survey	3rd survey	4th survey	5th survey
1	0	1	0	1	0
1	0	0	0	1	1
1	0	1	0	1	0
0	0	0	0	0	0
1	1	1	1	0	0
1	0	1	0	0	0
1	0	0	1	0	1
0	0	0	0	0	0
1	0	0	0	1	1
1	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0
0	0	0	0	0	0
1	0	0	1	1	0
1	1	1	1	1	0
0	0	0	0	0	0
1	0	0	0	1	1
1	1	1	1	1	0
0	0	0	0	0	0
0	0	0	0	0	0
1	1	0	0	0	1
1	0	0	0	1	1
0	0	0	0	0	0
1	1	1	0	1	0
1	0	1	1	0	0
1	1	1	0	0	0
1	0	1	0	0	1

Ahora finalmente visualizaremos gráficamente los datos de unos y ceros que hemos simulado para nuestro muestreo. Recuerde que se trata de valores de unos que representan si hemos detectado, o no hemos detectado al venado, en cada uno de los sitios de muestreo en cada una de las visitas.

```
par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
matplot(elev, jitter(y), pch = "*", frame.plot = FALSE, ylim = c(0, 1),
        xlab = "Altitud", ylab = "Detection/Nondetection (y)")
matplot(forest, jitter(y), pch = "*", frame.plot = FALSE, ylim = c(0, 1),
        xlab = "Forest cover", ylab = "Detection/Nondetection (y)")
matplot(temp, jitter(y), pch = "*", frame.plot = FALSE, ylim = c(0, 1),
        xlab = "Temperature", ylab = "Detection/Nondetection (y)")
hist(y, breaks = 50, col = "grey", ylim = c(0, 600), main = "",
     xlab = "Detection/Nondetection (y)")
```

Figura 5.4: Relación entre la ocupación observada (jittered) de venados (y) y las tres covariables estandarizadas. Altitud (A). Cobertura de bosque (B). Temperatura (C) y la frecuencia de distribución de la ocurrencia observada (y) en un set de datos de 60 sitios con 30 muestreos cada uno (D).

Hasta aquí hemos creado un set de datos donde la detección/no detección del venado esta negativamente correlacionado con la temperatura y positivamente relacionado con la cobertura de bosque. Hay una razón por la cual esta correlación entre variables es diferente. La ocurrencia por sitio, el objetivo de la inferencia ecológica, está afectado por la cobertura de bosque y la altitud, pero no por la temperatura, mientras que la probabilidad de detección, el parámetro caracterizando la medida del proceso de error cuando tomamos medidas de ocurrencia, es también afectado por la altitud y adicionalmente por la temperatura. Por lo tanto, como se puede notar, hay un gran reto para poder desenredar la razón de la variación espacio-temporal en la observación de los datos de detección/no detección, dado que pueden ser afectados por dos procesos totalmente diferentes: el ecológico y el observacional, que también la misma covariable puede afectar los dos procesos y que también pueden existir interacciones entre covariables.

Capítulo 6

Empacando todo en una función

Podría ser de mucha utilidad empacar todo lo que hemos hecho en una sola función que nos permita hacer lo mismo muchas veces repetidamente. Esto hará que podamos diseñar simulaciones de una forma más concisa y flexible y hace más transparente la generación de parámetros usados para generar datos. Asi que vamos a definir una función (que llamaremos data.fn) para generar el mismo tipo de datos que acabamos de crear, asignando argumentos a la función, tales como tamaño de la muestra, efectos de las covariables y direcciones y magnitudes de la interacción de los términos del error de detección y ocupación. Esto hará que nuestro código sea más flexible y eficiente.

```
################################
## The function starts here ###
################################

# Function definition with set of default values
data.fn <- function(M = 60, J = 30, mean.occupancy = 0.6,
                    beta1 = -2, beta2 = 2, beta3 = 1, mean.detection = 0.3,
                    alpha1 = -1, alpha2 = -3, alpha3 = 0, show.plot = TRUE){
# Function to simulate occupancy measurements replicated at M sites during J occasions.
# Population closure is assumed for each site.
# Expected occurrence may be affected by elevation (elev),
# forest cover (forest) and their interaction.
# Expected detection probability may be affected by elevation,
# temperature (temp) and their interaction.
# Function arguments:
#     M: Number of spatial replicates (sites)
#     J: Number of temporal replicates (occasions)
```

```
#     mean.occupancy: Mean occurrence at value 0 of occurrence covariates
#     beta1: Main effect of elevation on occurrence
#     beta2: Main effect of forest cover on occurrence
#     beta3: Interaction effect on occurrence of elevation and forest cover
#     mean.detection: Mean detection prob. at value 0 of detection covariates
#     alpha1: Main effect of elevation on detection probability
#     alpha2: Main effect of temperature on detection probability
#     alpha3: Interaction effect on detection of elevation and temperature
#     show.plot: if TRUE, plots of the data will be displayed;
#                set to FALSE if you are running simulations.

# Create covariates
elev <- runif(n = M, -1, 1)                          # Scaled elevation
forest <- runif(n = M, -1, 1)                        # Scaled forest cover
temp <- array(runif(n = M*J, -1, 1), dim = c(M, J)) # Scaled temperature

# Model for occurrence
beta0 <- qlogis(mean.occupancy)               # Mean occurrence on link scale
psi <- plogis(beta0 + beta1*elev + beta2*forest + beta3*elev*forest)
z <- rbinom(n = M, size = 1, prob = psi)      # Realised occurrence

# Plots
if(show.plot){
  par(mfrow = c(2, 2), cex.main = 1)
  devAskNewPage(ask = TRUE)
  curve(plogis(beta0 + beta1*x), -1, 1, col = "red", frame.plot = FALSE,
      ylim = c(0, 1), xlab = "Elevation", ylab = "psi", lwd = 2)
  plot(elev, psi, frame.plot = FALSE, ylim = c(0, 1), xlab = "Elevation",
     ylab = "")
  curve(plogis(beta0 + beta2*x), -1, 1, col = "red", frame.plot = FALSE,
      ylim = c(0, 1), xlab = "Forest cover", ylab = "psi", lwd = 2)
  plot(forest, psi, frame.plot = FALSE, ylim = c(0, 1), xlab = "Forest cover",
     ylab = "")
}

# Model for observations
y <- p <- matrix(NA, nrow = M, ncol = J)# Prepare matrix for y and p
alpha0 <- qlogis(mean.detection)        # mean detection on link scale
for (j in 1:J){                         # Generate counts by survey
   p[,j] <- plogis(alpha0 + alpha1*elev + alpha2*temp[,j] + alpha3*elev*temp[,j])
   y[,j] <- rbinom(n = M, size = 1, prob = z * p[,j])
}

# True and observed measures of 'distribution'
sumZ <- sum(z)                      # Total occurrence (all sites)
```

```
sumZ.obs <- sum(apply(y,1,max))    # Observed number of occ sites
psi.fs.true <- sum(z) / M          # True proportion of occ. sites in sample
psi.fs.obs <- mean(apply(y,1,max)) # Observed proportion of occ. sites in sample

# More plots
if(show.plot){
  par(mfrow = c(2, 2))
  curve(plogis(alpha0 + alpha1*x), -1, 1, col = "red",
      main = "Relationship p-elevation \nat average temperature",
      xlab = "Scaled elevation", frame.plot = F)
  matplot(elev, p, xlab = "Scaled elevation",
        main = "Relationship p-elevation\n at observed temperature",
        pch = "*", frame.plot = F)
  curve(plogis(alpha0 + alpha2*x), -1, 1, col = "red",
      main = "Relationship p-temperature \n at average elevation",
      xlab = "Scaled temperature", frame.plot = F)
  matplot(temp, p, xlab = "Scaled temperature",
        main = "Relationship p-temperature \nat observed elevation",
        pch = "*", frame.plot = F)
}

# Output
return(list(M = M, J = J, mean.occupancy = mean.occupancy,
            beta0 = beta0, beta1 = beta1, beta2 = beta2, beta3 = beta3,
            mean.detection = mean.detection,
            alpha0 = alpha0, alpha1 = alpha1, alpha2 = alpha2, alpha3 = alpha3,
            elev = elev, forest = forest, temp = temp,
            psi = psi, z = z, p = p, y = y, sumZ = sumZ, sumZ.obs = sumZ.obs,
            psi.fs.true = psi.fs.true, psi.fs.obs = psi.fs.obs))
}

################################
## The function ends  here  ###
################################
```

Una vez que hayamos definido la función y ejecutado su código, podremos llamarla repetidamente y enviar los resultados a la pantalla o asignarlos a un objeto en R. De forma tal que podamos usar el set de datos almacenado en el objeto para un análisis detallado.

```
# Run this part line by line, taking note of the meaning of the
# model in the comment and hitting Enter after each graph.
# Take in to account that if you do not override all the parameters
# with another value, the function will use the default values.

data.fn()                  # Execute function with default arguments
```

```
data.fn(show.plot = FALSE) # same, without plots
objeto1 <- data.fn(M = 30, J = 10, mean.occupancy = 0.5,
                   beta1 = -2, beta2 = 2, beta3 = 1,
                   mean.detection = 0.25, alpha1 = -1,
                   alpha2 = -3, alpha3 = 0, show.plot = TRUE) # Explicit defaults
```

Tal vez el uso más sencillo posible para esta función es experimentar de primera mano el error de muestreo: el cuál es la variabilidad natural de realizaciones repetidas (varios sets de datos) de nuestro proceso estocástico por el cual calculamos los set de datos. Vamos a simular 10.000 sets de datos del venado para ver como varían en términos del verdadero número de sitios ocupados (sumZ en el código) y el número de sitios en los que los venados fueron observados al menos una vez. Tenga en cuenta que los datos generados en las 10.000 tienen los parámetros por defecto en los parámetros mean.occupancy y mean.detection.

```
simrep <- 10000
trueSumZ <- obsSumZ <- numeric(simrep)
for(i in 1:simrep){
   if(i %% 1000 ==0 )                          # report progress
   cat("iter", i, "\n")
   data <- data.fn(M = 60,J = 3,show.plot = FALSE) # 60 sitios, 3 muestreos, p=0.3
   trueSumZ[i] <- data$sumZ
   obsSumZ[i] <- sum(apply(data$y, 1, max))
}
```

```
## iter 1000
## iter 2000
## iter 3000
## iter 4000
## iter 5000
## iter 6000
## iter 7000
## iter 8000
## iter 9000
## iter 10000
```

```
plot(sort(trueSumZ), ylim = c(min(obsSumZ), max(trueSumZ)), ylab = "", xlab = "Simulati
     col = "red", main = "True (red) and observed (blue) number of occupied sites")
points(obsSumZ[order(trueSumZ)], col = "blue")
```

Como ejercicio, cambie el código para generar otras vez las 10.000 simulaciones con una detección media (mean.detection =0.5), alta (mean.detection =0.8), y perfecta (mean.detection =1). Compare las gráficas resultantes.

Ahora podemos usar esta función para generar datos bajo diferentes esquemas de muestreo, variando el número de sitios y el número de muestreos repetidos.

Figura 6.1: Variabilidad natural (error de muestreo) del verdadero número de sitios ocupados (ordenados por tamaño) en color rojo y el número observado de sitios ocupados (en azul). El número de sitios observados/total también se conoce como la ocupación ingenua o (naïve) de la ocurrencia de los venados en 60 sitios en la simulación. El ancho del área azul representa el error inducido por la detección imperfecta. Note la importancia de tener en cuenta este error para tener una mejor idea de la ocupación.

Así como también bajo diferentes características ecológicas y de detección, y considerando posibles interacciones entre covariables.

```
# Run this part line by line, taking note of the meaning of the
# model in the comment and hitting Enter after each graph.
# Take in to account that if you do not override all the parameters
# with another value, the function will use the default values.

data.fn(J = 1, show.plot = T)  # Only 1 survey (no temporal replicate)
data.fn(J = 2, show.plot = T)  # Only 2 surveys (sites)
data.fn(M = 5, J = 3)          # Only 5 sites, 3 counts (repeted visits)
data.fn(M = 1, J = 100)        # No spatial replicates, but 100 counts
data.fn(M = 1000, J = 100)     # Very intensive sampling. 1000 sites, 100 visits

data.fn(mean.occupancy = 0.6,   # psi = 0.6 and
        mean.detection = 1,     # p = 1 (perfect detection!!!)
        show.plot = T)

data.fn(mean.occupancy = 0.95,  # psi = 1 a really coomon sp.
        mean.detection = 1,     # p = 1 (perfect detection!!!)
        show.plot = T)

data.fn(mean.occupancy = 0.05,  # psi = 0.05 a really rare sp.
        mean.detection = 0.05,  # p = 0.05 and very hard to detect !!!
        show.plot = T)

data.fn(beta3 = 1.5, show.plot = TRUE) # With interaction elev-temp on p

data.fn(mean.occupancy = 0.6, beta1 = -2, beta2 = 2, beta3 = 1,
        mean.detection = 0.1, show.plot = TRUE)  # p = 1 (low detectability)

data.fn(M = 267, J = 5, mean.occupancy = 0.6, beta1 = 0, beta2 = 0, beta3 = 0,
        mean.detection = 0.4, alpha1 = 0, alpha2 = 0, alpha3 = 0, show.plot = TRUE)
# Simplest case with occupancy (0.6) and detection (0.4) constant, no covariate effects
# observe betas = 0, and alphas = 0. This correspond to a kind of null model.
```

FELICITACIONES!!!, si llego hasta acá, y si entendió la

simulación de datos y su procedimiento, entonces Ud. entendió totalmente el modelo básico de ocupación, el cual es la piedra angular del muestreo y monitoreo biológico moderno.

Capítulo 7

Análisis de ocupación, metodo ML

Ya que hemos entendido como funcionan e interactúan los dos procesos; el ecológico y el observacional para producir los datos de ocupación. Luego de generar varios sets de datos, ahora solo nos resta analizarlos. La forma más directa e intuitiva es usar la función occu del paquete unmarked (**?**). Posteriormente podremos usar un modelo de tipo bayesiano en el lenguaje BUGS para analizar los mismos datos y al final comparar cuál de los dos estimadores, Máxima Verosimilitud o Bayesiano, se acerca más a los parámetros verdaderos.

7.0.1 Limpiando la memoria de R

Antes de continuar, y como ya hemos generado una gran cantidad de datos y modelos en los pasos previos, vamos a borrar la memoria de R antes de comenzar. Esto lo hacemos con el comando:

```
rm(list = ls())
```

Una vez hayamos hecho esto debemos volver a correr el código de la función que genera los datos que hemos creado el Capítulo **??**.

7.1 Generando los datos

Esta vez recurriremos a un diseño tipo TEAM (http://www.teamnetwork.org) con 60 sitios de muestreo y 30 visitas repetidas, que equivalen a los 30 días en que las cámaras permanecen activas en campo. De nuevo nuestra especie es el venado de cola blanca. Para este ejemplo asumiremos que la detección es 0.6, la

ocupación 0.8 y las interacciones son mucho más sencillas con la altitud como la única covariable que explica la ocupación. Sin embargo para la detección hay una relación más compleja, asumiendo que hay una leve interacción entre las covariables de la observación. Para la observación la altitud y temperatura interactúan entre sí. También observe como la altitud influye en direcciones opuestas con un signo positivo en la altitud para la detección y negativo para la ocupación.

```
# Data generation
# Lets build a model were elevation explain occupancy and p has interactions
datos2<-data.fn(M = 60, J = 30, show.plot = FALSE,
                mean.occupancy = 0.8, beta1 = -1.5, beta2 = 0, beta3 = 0,
                mean.detection = 0.6, alpha1 = 2, alpha2 = 1, alpha3 = 1.5
                )

# Function to simulate occupancy measurements replicated at M sites during J occasions
# Population closure is assumed for each site.
# Expected occurrence may be affected by elevation (elev),
# forest cover (forest) and their interaction.
# Expected detection probability may be affected by elevation,
# temperature (temp) and their interaction.
# Function arguments:
#     M: Number of spatial replicates (sites)
#     J: Number of temporal replicates (occasions)
#     mean.occupancy: Mean occurrence at value 0 of occurrence covariates
#     beta1: Main effect of elevation on occurrence
#     beta2: Main effect of forest cover on occurrence
#     beta3: Interaction effect on occurrence of elevation and forest cover
#     mean.detection: Mean detection prob. at value 0 of detection covariates
#     alpha1: Main effect of elevation on detection probability
#     alpha2: Main effect of temperature on detection probability
#     alpha3: Interaction effect on detection of elevation and temperature
#     show.plot: if TRUE, plots of the data will be displayed;
#                set to FALSE if you are running simulations.


#To make the objects inside the list directly accessible to R, without having to addre
#them as data$C for instance, you can attach datos2 to the search path.

attach(datos2)          # Make objects inside of 'datos2' accessible directly

#Remember to detach the data after use, and in particular before attaching a new data
#object, because more than one data set attached in the search path will cause confusi

# detach(datos2)          # Make clean up
```

7.2 Poniendo los datos en unmarked

Unmarked es el paquete de R que usamos para analizar los datos de ocupación (**?**). Para lograr esto debemos primero preparar los datos y juntarlos en un objeto de tipo unmarkedFrame. En este caso usamos la función unmarkedFrameOccu que es específica para análisis de ocupación de una sola época o estación. Más sobre unkarked en: https://sites.google.com/site/unmarkedinfo/home

```
library(unmarked)
siteCovs <- as.data.frame(cbind(forest,elev)) # occupancy covariates
obselev<-matrix(rep(elev,J),ncol = J) # make elevetion per observation
obsCovs <- list(temp= temp,elev=obselev) # detection covariates

# umf is the object joining observations (y), occupancy covariates (siteCovs)
# and detection covariates (obsCovs). Note that obsCovs should be a list of
# matrices or dataframes.

umf <- unmarkedFrameOccu(y = y, siteCovs = siteCovs, obsCovs = obsCovs)
```

El paquete unmarked nos permite ver gráficamente como están dispuestos los datos en los sitios de muestreo con la función plot.

```
plot (umf)
```

Figura 7.1: Inspección grafica del objeto umf.

7.3 Ajustando los modelos

El siguiente paso es ajustar los modelos que se requerían variando las co-variables. Esto se logra con la función occu().

Tenga en cuenta que en el proceso de construcción de modelos su modelo debe tener un significado biológico.

```
# detection first, occupancy next
fm0 <- occu(~1 ~1, umf) # Null model
fm1 <- occu(~ elev ~ 1, umf) # elev explaining detection
fm2 <- occu(~ elev ~ elev, umf) # elev explaining detection and occupancy
fm3 <- occu(~ temp ~ elev, umf)
fm4 <- occu(~ temp ~ forest, umf)
fm5 <- occu(~ elev + temp ~ 1, umf)
fm6 <- occu(~ elev + temp + elev:temp ~ 1, umf)
fm7 <- occu(~ elev + temp + elev:temp ~ elev, umf)
fm8 <- occu(~ elev + temp + elev:temp ~ forest, umf)
```

7.4 Selección de modelos

Unmarked permite hacer selección de modelos basándose en el AIC de cada uno. De forma tal que el menor AIC es el modelo más parsimonioso de acuerdo a nuestros datos (**?**).

```
models <- fitList( # here e put names to the models
  'p(.)psi(.)'                          = fm0,
  'p(elev)psi(.)'                       = fm1,
  'p(elev)psi(elev)'                    = fm2,
  'p(temp)psi(elev)'                    = fm3,
  'p(temp)psi(forest)'                  = fm4,
  'p(temp+elev)psi(.)'                  = fm5,
  'p(temp+elev+elev:temp)psi(.)'        = fm6,
  'p(temp+elev+elev:temp)psi(elev)'     = fm7,
  'p(temp+elev+elev:temp)psi(forest)' = fm8)

modSel(models) # model selection procedure
```

```
##                                    nPars     AIC  delta
## p(temp+elev+elev:temp)psi(elev)        6 1630.77   0.00
## p(temp+elev+elev:temp)psi(.)           5 1636.27   5.49
## p(temp+elev+elev:temp)psi(forest)      6 1638.24   7.47
## p(temp+elev)psi(.)                     4 1672.55  41.78
## p(elev)psi(elev)                       4 1741.34 110.57
## p(elev)psi(.)                          3 1746.83 116.06
## p(temp)psi(elev)                       4 1898.56 267.79
## p(temp)psi(forest)                     4 1906.03 275.26
## p(.)psi(.)                             2 1969.25 338.48
##                                      AICwt cumltvWt
## p(temp+elev+elev:temp)psi(elev)   9.2e-01     0.92
## p(temp+elev+elev:temp)psi(.)      5.9e-02     0.98
## p(temp+elev+elev:temp)psi(forest) 2.2e-02     1.00
## p(temp+elev)psi(.)                7.8e-10     1.00
## p(elev)psi(elev)                  9.0e-25     1.00
## p(elev)psi(.)                     5.8e-26     1.00
## p(temp)psi(elev)                  6.5e-59     1.00
## p(temp)psi(forest)                1.6e-60     1.00
## p(.)psi(.)                        2.9e-74     1.00
```

7.5 Predicción en graficas

El modelo con menor AIC puede ser usado para predecir resultados esperados de acuerdo a un nuevo set de datos. Por ejemplo, uno podría preguntar

la abundancia de venados que se espera encontrar en un sitio con mayor altitud. Las predicciones también son otra forma de presentar los resultados de un análisis. Aquí ilustraremos como se ve la predicción de ψ y p sobre el rango de las covariables estudiadas. Note que estamos usando covariables estandarizadas. Si estuviéramos usando covariables en su escala real, tendríamos que tener en cuenta que hay que transformarlas usando la media y la desviación estándar.

Antes de usar el modelo para predecir es buena idea verificar los parámetros del modelo y sus errores, posteriormente verificar gráficamente que el modelo ajusta bien con la función parboot, la cual hace un remuestreo del modelo. Esta grafica se interpreta como que el modelo tiene buen ajuste, cuando la media (línea punteada) está entre los intervalos del histograma. Si la línea se encuentra muy lejos del histograma el modelo no podría ser bueno para predecir.

```
summary(fm7) # see the model parameters
```

```
##
## Call:
## occu(formula = ~elev + temp + elev:temp ~ elev, data = umf)
##
## Occupancy (logit-scale):
##             Estimate    SE     z  P(>|z|)
## (Intercept)     1.41 0.366  3.85 0.000117
## elev           -1.63 0.656 -2.48 0.013082
##
## Detection (logit-scale):
##             Estimate     SE     z  P(>|z|)
## (Intercept)     0.53 0.0698  7.60 3.04e-14
## elev            1.80 0.1277 14.10 3.56e-45
## temp            1.16 0.1209  9.57 1.08e-21
## elev:temp       1.33 0.2211  6.00 1.96e-09
##
## AIC: 1630.771
## Number of sites: 60
## optim convergence code: 0
## optim iterations: 31
## Bootstrap iterations: 0
```

```
pb <- parboot(fm7, nsim=250, report=10) # goodness of fit
```

```
## t0 = 409.3793
```

```
plot (pb) # plot goodness of fit
```

Figura 7.2: Evaluación grafica del ajuste del modelo fm7.

Ahora que sabemos que nuestro mejor modelo tiene buen ajuste, podemos usarlo

para predecir la ocupación en el rango de la altitud para ver su comportamiento en una gráfica.

```
elevrange<-data.frame(elev=seq(min(datos2$elev),max(datos2$elev),length=100)) # newdat
pred_psi <-predict(fm7,type="state",newdata=elevrange,appendData=TRUE)
plot(Predicted~elev, pred_psi,type="l",col="blue",
       xlab="elev",
       ylab="psi")
lines(lower~elev, pred_psi,type="l",col=gray(0.5))
lines(upper~elev, pred_psi,type="l",col=gray(0.5))
```

Figura 7.3: Grafica de la ocupación con respecto a la elevación.

7.6 Predicción espacialmente explicita

Podemos también usar el mejor modelo para predecir de forma espacialmente explicita sí tenemos los mapas. Como ilustración vamos a construir mapas simulados para cada una de nuestras covariables. Los mapas surgen de un patrón aleatorio de puntos con distribución Poisson. Luego estos puntos los convertimos en una superficie interpolada.

```
# lets make random maps for the three covariates
library(raster)
library(spatstat)
set.seed(24) # Remove for random simulations

# CONSTRUCT ANALYSIS WINDOW USING THE FOLLOWING:
xrange=c(-2.5, 1002.5)
yrange=c(-2.5, 502.5)
window<-owin(xrange, yrange)

# Build maps from random points and interpole in same line
elev   <- density(rpoispp(lambda=0.6, win=window)) #
forest <- density(rpoispp(lambda=0.2, win=window)) #
temp   <- density(rpoispp(lambda=0.5, win=window)) #

# Convert covs to raster and Put in the same stack
mapdata.m<-stack(raster(elev),raster(forest), raster(temp))
names(mapdata.m)<- c("elev", "forest", "temp") # put names to raster

# lets plot the covs maps
plot(mapdata.m)
```

Una vez tenemos nuestros mapas de covariables, los usamos para predecir con

Figura 7.4: Mapa simulado de elevación, bosque y temperatura.

el mejor modelo. De esta forma podemos tener un mapa con predicciones de la ocupación y la probabilidad de detección.

```
# make the predictions
predictions_psi <- predict(fm7, type="state", newdata=mapdata.m) # predict psi
```

```
##   doing row 1000 of 16384
##   doing row 2000 of 16384
##   doing row 3000 of 16384
##   doing row 4000 of 16384
##   doing row 5000 of 16384
##   doing row 6000 of 16384
##   doing row 7000 of 16384
##   doing row 8000 of 16384
##   doing row 9000 of 16384
##   doing row 10000 of 16384
##   doing row 11000 of 16384
##   doing row 12000 of 16384
##   doing row 13000 of 16384
##   doing row 14000 of 16384
##   doing row 15000 of 16384
##   doing row 16000 of 16384
```

```
predictions_p   <- predict(fm7, type="det",   newdata=mapdata.m) # predict p
```

```
##   doing row 1000 of 16384
##   doing row 2000 of 16384
##   doing row 3000 of 16384
##   doing row 4000 of 16384
##   doing row 5000 of 16384
##   doing row 6000 of 16384
##   doing row 7000 of 16384
##   doing row 8000 of 16384
##   doing row 9000 of 16384
##   doing row 10000 of 16384
##   doing row 11000 of 16384
##   doing row 12000 of 16384
##   doing row 13000 of 16384
##   doing row 14000 of 16384
##   doing row 15000 of 16384
##   doing row 16000 of 16384
```

```
# put in the same stack
predmaps<-stack(predictions_psi$Predicted,predictions_p$Predicted)
```

```
names(predmaps)<-c("psi_predicted", "p_predicted") # put names
plot(predmaps)
```

Figura 7.5: Modelos de ocupación y detección espacialmente explícitos.

Capítulo 8

Análisis Bayesiano

En esta parte vamos a estimar los mismos parámetros de un modelo igual al "mejor modelo" el cual fue seleccionado en el procedimiento de selección de modelos de unmarked en el capitulo anterior. Recordemos que este modelo tiene beta1 y alpha1, alpha2, alpha3. Los parámetros que estimaremos con el método Bayesiano los vamos a comparar con los parámetros que ya estimamos con ML en unmarked y también los compararemos con los parámetros reales que definimos al establecer los datos (datos2) con la función data.fn, para ver cual de los dos métodos de estimación (ML o Bayesiano) se acerca más a los parámetros reales.

8.1 Generando los datos

De nuevo usaremos un diseño tipo TEAM (http://www.teamnetwork.org) con 60 sitios de muestreo y 30 visitas repetidas, que equivalen a los 30 días en que las camaras permenecen activas en campo. Nuestra especie sigue siendo la misma, el venado de cola blanca. Para este ejemplo asumiremos que la detección es 0.6, la ocupacion 0.8 y las interacciones son sencillas con la altitud como la unica covariable que explica la ocupación. Pero para la deteccion hay una relación más compleja, asumiendo que hay una leve interacción entre las covariables de la observación. Para la observación la altitud y temperatura interactuan entre si. Tambien observe como la altitud influye en direcciones opuestas con un signo positivo en la altitud para la detección y negativo para la ocupación.

```
# ### Generate a new data set or use the same
# # ************************************
# set.seed(148)
# data <- data.fn(show.plot = T)    # Default arguments
# str(data)                         # Look at the object
# we are oing to use the data from datos2 object
```

```
### Fit same model with JAGS, using library jagsUI
# ***********************************************
# Bundle data
win.data <- list(y = datos2$y,
                 M = nrow(datos2$y),
                 J = ncol(datos2$y),
                 elev = datos2$elev,
                 forest = datos2$forest,
                 temp = datos2$temp)
# str(win.data)


# # Specify model in BUGS language
# sink("model22.txt")
# cat("
# model {
#
# # Priors
# mean.p ~ dunif(0, 1)         # Detection intercept on prob. scale
# alpha0 <- logit(mean.p)      # same on logit scale
# mean.psi ~ dunif(0, 1)       # Occupancy intercept on prob. scale
# beta0 <- logit(mean.psi)     # same on logit scale
# for(k in 1:3){               # 2 detection covariates + 1 interact
#     alpha[k] ~ dnorm(0, 0.01) # Covariates on logit(detection)
# #   alpha[k] ~ dnorm(0, 0.05) # Covariates on logit(detection)
# #   alpha[k] ~ dunif(-10, 10) # Covariates on logit(detection)
# }
#
# for(k in 1:1){                 # 2 occupancy covariates + 1 interact
#     beta[k] ~ dnorm(0, 0.01)  # Covariates on logit(occupancy)
# #   beta[k] ~ dnorm(0, 0.05)  # Covariates on logit(occupancy)
# #   beta[k] ~ dunif(-10, 10)  # Covariates on logit(occupancy)
# }
#
# # Translation of the occupancy parameters in unmarked into those for BUGS:
# # (Intercept)         (beta0 in BUGS)
# # elev                (beta[1])
# # forest              (beta[2])
# # temp                (beta[3])
# # elev:forest         (beta[4])
# # elev:temp           (beta[5])
# # forest:temp         (beta[6])
# # elev:forest:temp    (beta[7])
#
#
```

```
# # Likelihood
# for (i in 1:M) {
#   # True state model for the partially observed true state
#   z[i] ~ dbern(psi[i])                      # True occupancy z at site i
#   logit(psi[i]) <- beta0 +                  # occupancy (psi) intercept
#     beta[1] * elev[i] #+                     # elev
#     #beta[2] * forest[i] #+                  # forest
#     #beta[3] * elev[i] * forest[i]          # elev:forest
#     #beta[4] * elev[i] * temp[i] +          # elev:temp
#     #beta[5] * temp[i] +                    # temp
#     #beta[6] * forest[i] * temp[i] +        # forest:temp
#     #beta[7] * elev[i] * forest[i] * temp[i]   # elev:forest:temp
#
#    for (j in 1:J) {
#       # Observation model for the actual observations
#       y[i,j] ~ dbern(p.eff[i,j])      # Detection-nondetection at i and j
#       p.eff[i,j] <- z[i] * p[i,j]
#       logit(p[i,j]) <- alpha0 +             # detection (p) intercept
#          alpha[1] * elev[i] +               # effect of elevation on p
#          alpha[2] * temp[i,j] +             # effect of temp on p
#          alpha[3] * elev[i] * temp[i,j]     # effect of elev:temp on p
#    }
# }
#
# # Derived quantities
# sumZ <- sum(z[])      # Number of occupied sites among those studied
# occ.fs <- sum(z[])/M  # proportion of occupied sites among those studied
# logit.psi <- beta0    # For comparison with unmarked
# logit.p <- alpha0     # For comparison with unmarked
# }
# ",fill = TRUE)
# sink()


library(jagsUI)
# library(R2jags)
# Initial values
zst <- apply(datos2$y, 1, max)
inits <- function(){list(z = zst,
                         mean.psi = runif(1),
                         mean.p = runif(1),
                         alpha = rnorm(3), # adjust here
                         beta = rnorm(1))} # adjust here

# Parameters monitored
```

```
params <- c("sumZ", "occ.fs", "logit.psi", "logit.p", "alpha", "beta")

# MCMC settings
# ni <- 100000   ;   nt <- 10   ;   nb <- 1000   ;   nc <- 3
 ni <- 10000   ;   nt <- 10   ;   nb <- 500   ;   nc <- 3

# Call JAGS from R (ART 260 sec with norm(), 480 with unif(-10,10))
# and summarize posteriors
system.time(out22 <- jags(win.data, inits, parameters.to.save = params,
                      model.file = "D:/BoxFiles/Box Sync/CodigoR/Toshiba/IntroOccuBook/
                          n.chains = nc,
                          n.thin = nt,
                          n.iter = ni,
                          n.burnin = nb,
                          parallel = T))
```

```
##
## Processing function input.......
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
##    user  system elapsed
##    0.08    0.02  225.25
```

```
# See model diagnistics and convergence
library(mcmcplots)
library(ggmcmc)
fit22.mcmc <- as.mcmc.list(out22$samples)
bayes.mod.fit.gg <- ggs(fit22.mcmc) #convert to ggmcmc
ggs_running(bayes.mod.fit.gg)# check if chains approach target distrib.


# denplot(fit22.mcmc, parms = c("beta",
#                               "alpha[1]", "alpha[2]", "alpha[3]",
#                               "logit.psi", "logit.p" ))
# traplot(fit22.mcmc)
# ggs_density(bayes.mod.fit.gg)
```

```
xyplot(out22)        # assess within-chain convergence
```

```
densityplot(out22)  # shape of the posterior distribution
```

```
# see model result and estimates
print(out22, 3)
```

```
## JAGS output for model 'D:/BoxFiles/Box Sync/CodigoR/Toshiba/IntroOccuBook/bookdown-demo-master
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 100 iterations (sufficient),
## burn-in = 500 iterations and thin rate = 10,
## yielding 2850 total samples from the joint posterior.
## MCMC ran in parallel for 3.753 minutes at time 2020-04-20 00:43:14.
##
##               mean    sd     2.5%      50%    97.5% overlap0
## sumZ        46.029 0.168   46.000   46.000   47.000    FALSE
## occ.fs       0.767 0.003    0.767    0.767    0.783    FALSE
## logit.psi    1.173 0.324    0.562    1.163    1.825    FALSE
## logit.p      0.408 0.074    0.263    0.407    0.560    FALSE
## alpha[1]     2.147 0.152    1.853    2.143    2.445    FALSE
## alpha[2]     0.902 0.126    0.657    0.903    1.155    FALSE
## alpha[3]     1.686 0.247    1.208    1.682    2.180    FALSE
## beta        -1.475 0.632   -2.769   -1.458   -0.329    FALSE
## deviance  1589.362 3.508 1585.504 1588.480 1599.696    FALSE
##               f  Rhat n.eff
## sumZ      1.000 1.000  2850
## occ.fs    1.000 1.000  2850
## logit.psi 1.000 1.000  2850
## logit.p   1.000 1.001  2372
## alpha[1]  1.000 1.000  2850
## alpha[2]  1.000 1.000  2850
## alpha[3]  1.000 1.000  2850
## beta      0.995 1.000  2850
## deviance  1.000 1.000  2850
##
## 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 = 6.2 and DIC = 1595.517
## DIC is an estimate of expected predictive error (lower is better).
```

```
# store in tmp coefficients from best ML model
tmp <- summary(fm7)
```

```
##
## Call:
## occu(formula = ~elev + temp + elev:temp ~ elev, data = umf)
##
## Occupancy (logit-scale):
##             Estimate    SE     z  P(>|z|)
## (Intercept)     1.18 0.326  3.62 0.000291
## elev           -1.41 0.624 -2.26 0.023548
##
## Detection (logit-scale):
##             Estimate     SE     z  P(>|z|)
## (Intercept)    0.407 0.0735  5.54 3.03e-08
## elev           2.140 0.1494 14.32 1.66e-46
## temp           0.900 0.1262  7.14 9.68e-13
## elev:temp      1.680 0.2514  6.68 2.34e-11
##
## AIC: 1656.533
## Number of sites: 60
## optim convergence code: 0
## optim iterations: 37
## Bootstrap iterations: 0
```

```
modestimates <- cbind(rbind(tmp$state[1:2], tmp$det[1:2]),
                Post.mean = out22$summary[c(3, 8, 4:7), 1],
                Post.sd   = out22$summary[c(3, 8, 4:7), 2] )

# fix the(logit-scale) in unmarked
modestimates[1,1]<- plogis(modestimates[1,1])
modestimates[3,1]<- plogis(modestimates[3,1])

# fix the(logit-scale) in Bayes in logit.psi  logit.p
modestimates[1,3]<- plogis(modestimates[1,3])
modestimates[3,3]<- plogis(modestimates[3,3])

# get real values from datos2 object
real<- rbind(datos2$mean.occupancy, datos2$beta1, datos2$mean.detection,
             datos2$alpha1, datos2$alpha2, datos2$alpha3)
```

8.2 Comparando los valores reales y los estimados de ML y Bayesiano

Veamos qué tan cerca están los estimados de los valores reales, comparando el valor real con el estimado de Máxima Verosimilitud (columnas 2 y 3) y el estimado Bayesiano (columnas 4 y 5).

```
### see if the values are close to real values
compare <- cbind(real, modestimates) # put both in same table
# put names to rows
rownames(compare) <- c("psi","beta","p","alpha1","alpha2", "alpha3")

# print comparing table
library(knitr)
kable(compare)
```

	real	Estimate	SE	Post.mean	Post.sd
psi	0.8	0.7653197	0.3262574	0.7637476	0.3244823
beta	-1.5	-1.4129006	0.6239572	-1.4747848	0.6319201
p	0.6	0.6004304	0.0735178	0.6005410	0.0744655
alpha1	2.0	2.1395071	0.1494144	2.1470247	0.1515634
alpha2	1.0	0.9003900	0.1261927	0.9021164	0.1257829
alpha3	1.5	1.6801587	0.2513946	1.6858529	0.2468270

Capítulo 9

Información de la sesión de R y los paquetes usados

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 14393)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Spanish_Colombia.1252
## [2] LC_CTYPE=Spanish_Colombia.1252
## [3] LC_MONETARY=Spanish_Colombia.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=Spanish_Colombia.1252
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils
## [6] datasets  methods   base
##
## other attached packages:
##  [1] ggmcmc_1.4.1        ggplot2_3.2.1
##  [3] tidyr_1.0.2         dplyr_0.8.5
##  [5] mcmcplots_0.4.3     coda_0.19-3
##  [7] jagsUI_1.5.1        spatstat_1.63-3
##  [9] rpart_4.1-15        nlme_3.1-144
## [11] spatstat.data_1.4-3 raster_3.0-12
## [13] sp_1.4-0            unmarked_0.13-2
```

```
## [15] Rcpp_1.0.3          lattice_0.20-38
## [17] knitr_1.28          bookdown_0.18
##
## loaded via a namespace (and not attached):
##  [1] tufte_0.5             sfsmisc_1.1-5
##  [3] splines_3.6.1         StanHeaders_2.21.0-1
##  [5] assertthat_0.2.1      highr_0.8
##  [7] stats4_3.6.1          yaml_2.2.1
##  [9] pillar_1.4.3          glue_1.3.1
## [11] digest_0.6.24         RColorBrewer_1.1-2
## [13] polyclip_1.10-0       colorspace_1.4-1
## [15] htmltools_0.4.0       Matrix_1.2-18
## [17] plyr_1.8.5            pkgconfig_2.0.3
## [19] rstan_2.19.3          purrr_0.3.3
## [21] scales_1.1.0          processx_3.4.2
## [23] rjags_4-10            tensor_1.5
## [25] spatstat.utils_1.17-0 tibble_2.1.3
## [27] mgcv_1.8-31           farver_2.0.3
## [29] ellipsis_0.3.0        withr_2.1.2
## [31] lazyeval_0.2.2        cli_2.0.1
## [33] magrittr_1.5          crayon_1.3.4
## [35] deldir_0.1-25         evaluate_0.14
## [37] ps_1.3.2              GGally_1.5.0
## [39] fansi_0.4.1           MASS_7.3-51.5
## [41] pkgbuild_1.0.6        rsconnect_0.8.16
## [43] tools_3.6.1           loo_2.2.0
## [45] prettyunits_1.1.1     lifecycle_0.1.0
## [47] matrixStats_0.55.0    stringr_1.4.0
## [49] munsell_0.5.0         callr_3.4.2
## [51] packrat_0.5.0         compiler_3.6.1
## [53] tinytex_0.19          rlang_0.4.4
## [55] grid_3.6.1            rstudioapi_0.11
## [57] goftest_1.2-2         labeling_0.3
## [59] rmarkdown_2.1         gtable_0.3.0
## [61] codetools_0.2-16      reshape_0.8.8
## [63] inline_0.3.15         abind_1.4-5
## [65] reshape2_1.4.3        R6_2.4.1
## [67] gridExtra_2.3         denstrip_1.5.4
## [69] rgdal_1.4-8           stringi_1.4.6
## [71] vctrs_0.2.2           png_0.1-7
## [73] tidyselect_1.0.0      xfun_0.12
```