Estadística Computacional

Examen Final

Autores:
Omar Díaz

Supervisor:
Dr. Teresa Ortiz

18 de diciembre de 2014

Contents

Listings

1 Instrucciones

En las siguientes preguntas describe tus procedimientos y escribe las respuestas explícitamente (que no haya necesidad de correr código para obtenerlas). Incluye el código comentado.

2 Pregunta 1: Inferencia Gráfica

1. La base de datos **places** (Boyer y Savageau 1984) contiene ratings de varios aspectos de ciudades de EUA. El objetivo de este ejercicio es investigar si las variables en estos datos están asociadas, en particular se considera clima (Climate) y costo de vivienda (HousingCost). Valores bajos en clima implican temperaturas inconvenientes, puede ser mucho calor o mucho frío, mientras que valores altos corresponden a temperaturas más moderadas. Por su parte, valores altos en vivienda indican costos altos para una casa familiar simple.

   (a) ¿Qué relación esperarías entre las variables? Escribe la hipótesis nula.

   Debido a que las variables que se pretenden estudiar son el clima (Climate) y costo de vivienda (HousingCost), es lógico pensar que existe alguna relación entre ellas. Dicha relación puede ser que mientras mas agradable es el aspecto climatológico de alguna zona en particular, entonces el precio pudiera ser más elevado comparado con aquellas zonas cuyo clima tiende a ser extremoso en determinadas épocas del año. Ahora bien, como queremos saber si este supuesto tiene sentido, entonces debemos proponer como hipótesis nula que no existe ningún tipo de relación entre las variables, de tal manera que si la rechazamos con cierto nivel de significancia entonces nuestro supuesto inicial será válido.

   (b) Describe un método gráfico para probar tu hipótesis e implementalo. Genera 9 conjuntos de datos nulos y graficalos junto con los datos reales, escribe el nivel de significancia de la prueba y tus conclusiones.

   Para realizar un método gráfico para probar la hipótesis debemos esconder a los datos verdaderos entre 9 conjuntos de datos nulos, de tal suerte que si es posible identificar los datos entonces hay evidencia de que dichos datos son distintos a los datos nulos, rechazando así la hipótesis nula.

El conjunto de gráficas obtenido es el siguiente:

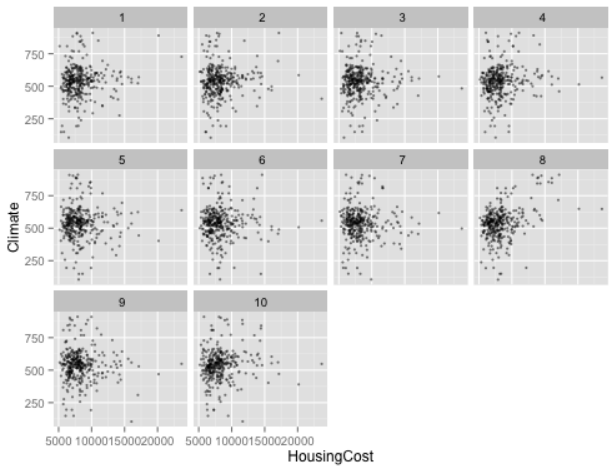

Figure 1: Método Gráfico 1

Debido a que los datos reales se encuentren escondidos entre 9 gráficas de datos nulos, la significancia de esta prueba es de $\frac{1}{10}$. Cabe mencionar que la manera en que se construyeron los conjuntos de datos nulos fue a partir de permutaciones de los valores de cada variable de manera aleatoria, de tal manera que si al permutarlos no es posible encontrar los datos originales entonces no es posible rechazar la hipótesis nula de no asociación entre dichas variables. Sin embargo, si se observa detenidamente el mosaico de gráficas es posible identificar en la figura 1 que la posición número 8 es considerablemente distinta a las demás. Por lo tanto, es posible rechazar la hipótesis nula de no relación entre las variables, confirmando así nuestro supuesto inicial acerca de la posible correlación de las variables.

A continuación se muestra el código de `R` que se utilizó para construir la prueba gráfica:

Listing 1: Inferencia Gráfica 1

```
places <- read.table("places.csv", header=TRUE, quote="\"")
places_ch <- places[,c("Climate","HousingCost")]
head(places_ch)
# valores bajos de clima implican climas inconvenientes
# valores altos en housing implican costos altos

# la hipotesis nula considera que no hay relacion entre los datos

# los datos originales se ven como siguen
```

```
ggplot(places_ch, aes(x = HousingCost, y = Climate)) +
  geom_point()

# inferencia por graficas
nrow(places_ch)

clim_null <- lineup(null_permute('HousingCost'), n = 10, places_ch)

ggplot(clim_null, aes(x = HousingCost, y = Climate)) +
  facet_wrap( ~ .sample) +
  geom_jitter(position = position_jitter(width = 0.1, height = 1),
              size = 1, alpha = 0.5)

decrypt("xrgp bViV NE njJNiNjE G")
```

3 Pregunta 2: Inferencia Gráfica

Para este ejercicio utilizaremos los datos de un estudio longitudinal de Singer y Willet 2003 (wages). En este estudio se visitó a hombres en edad laboral que habitan en EUA, se visitó a cada sujeto entre 1 y 13 veces, en cada visita se registraron las siguientes mediciones:

- id: identificador de sujeto
- hgc: grado de educación más alto completado
- lnw : logaritmo natural del salario
- exper: años de experiencia laboral

El objetivo del ejercicio es estudiar la relación entre salario y experiencia laboral por raza para aquellos sujetos cuyo año máximo de estudios completados es igual a 9, 10 u 11, estos son sujetos que abandonaron sus estudios durante preparatoria. Seguiremos un enfoque no paramétrico que consiste en ajustar un suavizador para cada grupo de raza (blanco, hispano o negro) como se muestra en la siguiente gráfica.

Listing 2: Instrucción de la gráfica

```
ggplot(wages_t, aes(x = exper, y = lnw)) +
  geom_point(alpha = 0.25, size = 2) +
  geom_smooth(aes(group = race, color = race), method = "loess", se = FALSE)
```

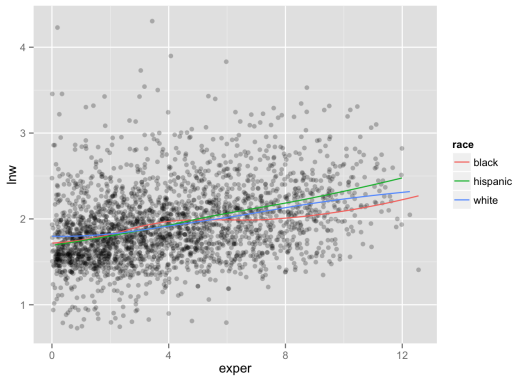

Figure 2: Relación salario-experiencia laboral por raza

Utilizaremos una prueba de hipótesis gráfica para determinar si existe una diferencia significativa entre las curvas.

3.1 Preparación de los Datos

Las consideraciones a tener para hacer el análisis son las siguientes:

- Selecciona los sujetos con grado de estudios completado igual a 9, 10 u 11.
- Elimina las observaciones donde el logaritmo del salario (lnw) es mayor a 3.5.
- Crea una variable correspondiente a raza, un sujeto es de raza hispana si la variable hispanic toma el valor 1, de raza negra si la variable black toma el valor 1 y de raza blanca si las dos anteriores son cero.
- Crea un subconjunto de la base de datos de tal manera que tengas el mismo número de sujetos distintos en cada grupo de raza. Nota: habrá el mismo número de sujetos en cada grupo pero el número de observaciones puede diferir pues los sujetos fueron visitados un número distinto de veces.

El código que genera esta nueva base de datos a partir de la cual se aplicará un modelo gráfico es el que sigue:

Listing 3: Preparación de los Datos

```
# preparacion de la informacion
wages <- read.csv("wages.csv")
wages <- subset(wages, hgc >= 9 & hgc <= 11 & lnw < 3.5)
wages$raza <- ifelse((wages$hispanic == 1), "hisp",
                     ifelse((wages$black==1), "black","white"))
```

```
6
7  # usamos el mismo numpero de individuos por raza
8  u.hisp <- unique(subset(wages, raza=="hisp",select=c(id,raza)))
9  u.black <- unique(subset(wages, raza=="black",select=c(id,raza)))
10 u.white <- unique(subset(wages, raza=="white",select=c(id,raza)))
11
12 ind <- c(sample(u.hisp$id,size=110,replace=FALSE),
13           sample(u.black$id,size=110,replace=FALSE),
14           sample(u.white$id,size=110,replace=FALSE))
15
16 wages_r <- wages[wages$id %in% ind, ]
```

3.2 Prueba de Hipótesis Visual

Para esta prueba el escenario nulo consiste en que no hay diferencia entre las razas. Para generar los datos nulos, la etiqueta de raza de cada sujeto se permuta, es decir, se reasigna la raza de cada sujeto de manera aleatoria (para todas las mediciones de un sujeto dado se reasigna una misma raza).

Se generaron 19 conjuntos de datos nulos y para cada uno ajusta una curva loess siguiendo la instrucción de la gráfica de arriba, después los datos nulos junto con los reales se colocan de manera aleatoria en una gráfica de paneles. La idea es realizar una prueba de hipótesis similar a la presentada en el ejercicio anterior, es decir, si es posible identificar que una de las gráficas es significativamente distinta y resulta ser aquella que muestra los valores observados, entonces es posible decir que existe diferencia entre el salario y experiencia laboral dependiendo de la raza.

La gráfica generada es la siguiente:

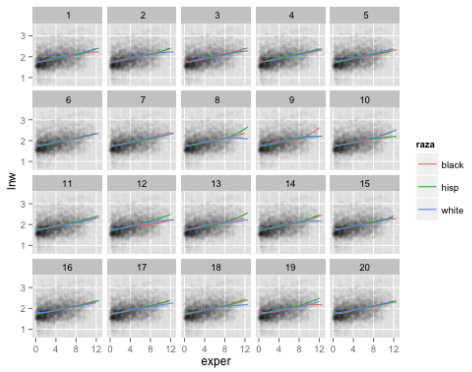

Figure 3: Panel de gráficas salario-experiencia laboral por raza

Una vez generada la gráfica se realiza la pregunta a 2 personas que no tienen la misma formación académica para dar fe y legalidad a la prueba. "Las siguientes 20 gráficas muestran suavizamientos de log(salarios) por años de experiencia laboral. Una de ellas usa datos reales y las otras 19 son datos nulos, generados bajo el supuesto de que no existe diferencia entre los subgrupos. ¿Cuál es la gráfica más distinta?"

En ambos casos, la respuesta fue la misma, la gráfica en la posición número 12 es aquella que parece proceder de una naturaleza distinta. Acertando con la gráfica generada por los datos reales.

En conclusión, la prueba de hipótesis visual es una herramienta muy útil cuando no es posible obtener estadísticos de prueba con una distribución familiar. Sin embargo, considero que en el ámbito profesional, podría ser una prueba de poco impacto en el sentido de tomar decisiones delicadas a partir de una prueba de esta naturaleza.

4 Pregunta 3: Relación entre Bootstrap e Inferencia Bayesiana

Consideremos el caso en que tenemos una única observación x proveniente de una distribución normal

$$x \sim N(\theta, 1) \tag{1}$$

Supongamos ahora que elegimos una distribución inicial Normal

$$\theta \sim N(0, \tau) \tag{2}$$

dando lugar a la distribución posterior (como vimos en la tarea)

$$\theta|x \sim N\left(\frac{x}{1+1/\tau}, \frac{1}{1+\tau}\right) \tag{3}$$

Ahora, entre mayor τ, más se concentra la posterior en el estimador de máxima verosimilitud $\hat{\theta} = x$. En el límite, cuando $\tau \to \infty$ obtenemos una inicial no-informativa (constante) y la distribución posterior

$$\theta|x \sim N(x, 1) \tag{4}$$

Esta posterior coincide con la distribución de bootstrap paramétrico en que generamos valores x^* de $N(x, 1)$, donde x es el estimador de máxima verosimilitud.

Lo anterior se cumple debido a que utilizamos un ejemplo Normal pero también se cumple aproximadamente en otros casos, lo que conlleva a una correspondencia entre el bootstrap paramétrico y la inferencia bayesiana. En este caso, la distribución bootstrap representa (aproximadamente) una distribución posterior no-informativa del parámetro de interés. Mediante la perturbación en los datos el bootstrap aproxima el efecto bayesiano de perturbar los parámetros con la ventaja de ser más simple de implementar (en muchos casos). Los detalles se pueden leer en The Elements of Statistical Learning de Hastie y Tibshirani.

Comparemos los métodos en otro problema con el fin de apreciar la similitud en los procedimientos:

Supongamos $x_1, \cdots, x_n \sim N(0, \sigma^2)$, es decir, los datos provienen de una distribución con media cero y varianza desconocida. Buscamos hacer inferencia del parámetro σ^2.

4.1 Bootstrap Paramétrico

Para realizar bootstrap paramétrico necesitamos conocer en principio la función de verosimilitud para posteriormente calcular el estimador de máxima verosimilitud de σ^2.

Sabemos la función de densidad de una normal es:

$$f(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{\sigma^2}\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{5}$$

la función de verosimilitud se define como

$$\begin{aligned} L(\sigma^2) &= \Pi_{i=1}^n f(x_i; \mu, \sigma^2) && (6) \\ &= (\sigma^2)^{\frac{-n}{2}} (2\pi)^{\frac{-n}{2}} e^{-\frac{\sum_{i=1}^n (x_i-\mu)^2}{2\sigma^2}} && (7) \end{aligned}$$

y la log-verosimilitud es entonces:

$$l(\sigma^2) = \frac{-n}{2} ln(\sigma^2) - \frac{-n}{2} ln(2\pi) - \frac{\sum_{i=1}^n (x_i - \mu)^2}{2\sigma^2} \tag{8}$$

Finalmente devirando respecto a σ^2 e igualando a cero se obtiene el estimador de máxima verosimilitud

$$\hat{\sigma^2} = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n} \tag{9}$$

A partir de la base de datos $\mathbf{X}$ obtenemos el estimador de máxima verosimilitud que es $\hat{\sigma^2} = 131.291$. Una vez que conocemos la estimación de máxima verosimilitud de la varianza, podemos conocer el error estándar de la estimación

a partir del bootstrap paramétrico y realizar un histograma de las replicaciones bootstrap.

La estimación a partir del método de bootstrap paramétrico es $\hat{\sigma^2} = 131.0694$ con un error estándar de $es = 15.197$. El histograma de las replicaciones se muestra a continuación:

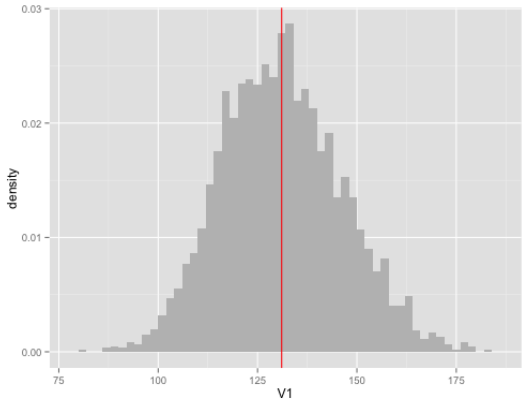

Figure 4: Histograma replicaciones bootstrap

El código es el siguiente:

Listing 4: Bootstrap Paramétrico

```
1
2  load("x.Rdata")
3
4  n<-length(x)
5
6  var <- (1 / n * sum((x - 0) ^ 2))
7  # 131.291
8
9  # Paso 1: calculamos la varianza
10 sigma2 <- 1 / n * sum((x - 0) ^ 2)
11
12 # Pasos 2 y 3
13 thetaBoot <- function(){
14   # Simular X_1^*,...X_N^* con distribucion N(mu_hat, sigma_hat^2)
15   x_boot <- rnorm(n, mean = 0, sd = sqrt(sigma2))
16   # Calcular mu*, sigma* y theta*
17   sigma2_boot <- 1 / n * sum((x_boot - 0) ^ 2)
18   sigma2_boot
19 }
20
21 # Paso 4 Repetimos B tresmil veces estimamos el error estandar
22 sims_boot <- rdply(3000, thetaBoot())
```

```
23
24 # sigma2 a partir de bootstrap
25 sigma2_b <- mean(sims_boot$V1)
26 sigma2_b
27
28 # erro estandar bootstrap
29 es_b <- sqrt(1 / 2999 * sum((sims_boot$V1 - sigma2) ^ 2))
30
31 # histograma de simulaciones bootstrap
32 ggplot(sims_boot, aes(x = V1)) +
33   geom_histogram(aes(y = ..density..), binwidth = 2, fill = "gray") +
34   geom_vline(xintercept = c(sigma2_b),
35              color = c("red"))
```

4.2 Análisis Bayesiano

Continuamos con el problema de hacer inferencia sobre σ^2. Para ello comenzamos especificando una distribución Gamma-Inversa como inicial. Los parámetros seleccionados fueron $shape = 1$ y $scale = 1$. La grafica de la distribución con dichos parámetros es la siguiente:

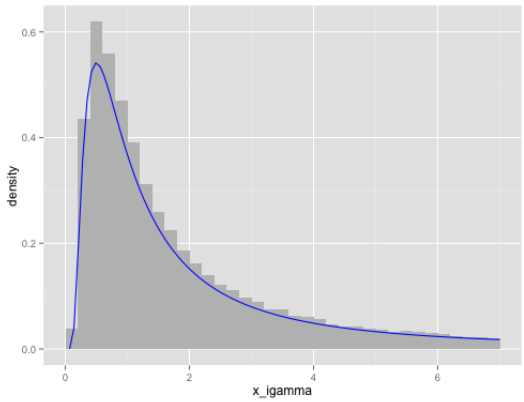

Figure 5: Histograma replicaciones bootstrap

Para poder hacer el análisis bayesiano es necesario obtener la distribución posterior, la cual se puede obtener de manera analítica.

Tenemos que encontrar

$$\begin{aligned} P(\sigma^2|x_i,\mu) &= P(x_i|\mu,\sigma^2)P(\sigma^2) \\ &= N(\mu,\sigma^2)Invgamma(\alpha_0,\beta_0) \end{aligned} \tag{10}$$

sea $\theta = \sigma^2$ entonces

$$
\begin{aligned}
P(\theta|x_i,\mu) &\alpha \Pi_{i=1}^{n}\left(\frac{1}{\sqrt{2\pi\theta}}e^{-\frac{(x_i-\mu)^2}{2\theta}}\right)\left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\theta^{-(\alpha_0+1)}e^{\frac{-\beta_0}{\theta}}\right)\\
&\alpha \Pi_{i=1}^{n}\left(\theta^{\frac{-1}{2}}e^{-\frac{(x_i-\mu)^2}{2\theta}}\right)\left(\theta^{-(\alpha_0+1)}e^{\frac{-\beta_0}{\theta}}\right) \qquad (11)\\
&= \theta^{\frac{-n}{2}}e^{-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\theta}}\theta^{-(\alpha_0+1)}e^{\frac{-\beta_0}{\theta}}\\
&= \theta^{-(\alpha_0+\frac{n}{2}+1)}e^{-\left(\frac{\beta_0}{\theta}+\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\theta}\right)}\\
&= \theta^{-(\alpha_0+\frac{n}{2}+1)}e^{-\left(\frac{\beta_0+\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2}}{\theta}\right)}
\end{aligned}
$$

(12)

Que se distribuye Gamma-Inversa con parametros $\alpha_1 = \alpha_0 + \frac{n}{2}$ y $\beta_1 = \beta_0 + \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2}$, es decir, se trata de una distribución conjugada.

Una vez que encontramos la distribución de la posterior, podemos hacer simular realizaciones de variables aleatorias con dicha distribución para así generar una distribución de la posterior a partir de un histograma, así como calcular el error estándar de la distribución.

El histograma de la distribución posterior es:

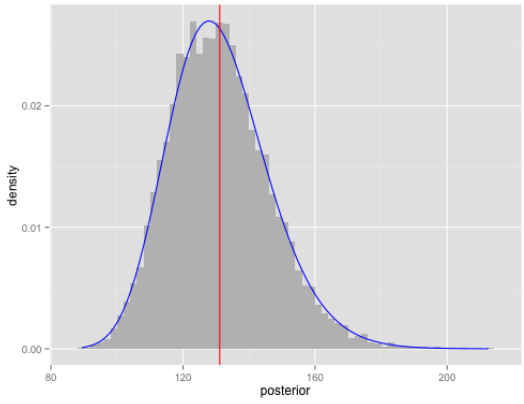

Figure 6: Histograma de simulaciones de la distribución posterior

Cuya media es $\hat{\sigma^2} = 131.1105$ y con error estándar de la distribución de 15.35968.

El código de esta sección es el siguiente:

Listing 5: Analisis Bayesiano

```
1
2  # definimos la seleccion de los parametros
3
4  shape0 <- 1
5  scale0 <- 1
6
7  # 1. simulamos valores provenientes de una distribucion gamma
8  x_gamma <- rgamma(100000, shape = shape0, rate = scale0)
9  # 2. invertimos los valores simulados
10 x_igamma <- 1 / x_gamma
11 x_igamma <- data.frame(x_igamma)
12
13 # grafica de la funcion de densidad de la gamma inversa
14 ggplot(x_igamma, aes(x = x_igamma)) +
15   geom_histogram(aes(y = ..density..), binwidth = .2, fill = "gray") +
16   stat_function(fun = dinvgamma, args = list(shape = shape0, scale = scale0),
17                 color = "blue") +
18   scale_x_continuous(limits = c(0, 7))
19
20
21
22 # calculo de la distribucion posterior
23 alpha1 <- shape0 + n/2
24 beta1 <- scale0 + sum((x - 0)^2)/2
25
26 posterior <- rinvgamma(10000, alpha1, beta1)
27
28 # valor de sigma2 estimado por analisis bayesianos
29 sigma2_bayes<- mean(posterior)
30 sigma2_bayes
31
32
33 # histograma posterior
34 ggplot(as.data.frame(posterior), aes(x = posterior)) +
35   geom_histogram(aes(y = ..density..), binwidth = 2, fill = "gray") +
36   stat_function(fun = dinvgamma, args = list(shape = alpha1, scale = beta1),
37                 color = "blue") +
38   geom_vline(xintercept = c(sigma2_bayes),
39              color = c("red"))
40
41 # error estandar
42 es_bayes <- sqrt(var(posterior))
43 es_bayes
```

4.2.1 Inicial no informativa

Ahora se realiza el mismo ejercicio pero considerando una distribución inicial no informativa, en este caso se trata de una $U(0.1, 300)$ y utilizamos JAGS (la función dunif indica una distribución uniforme) para obtener una muestra de simulaciones de la distribución posterior, recuerdemos que en JAGS la distribución Normal está parametrizada en términos de la precisión $\nu = 1/\sigma^2$. Una

vez tomada esta consideración, se debe obtener la distribución posterior a partir de un muestrador de Gibbs y para ello usaremos el paquete R2jags de R. El código que se implemento en JAGS para obtener simulaciones de la posterior es el siguiente:

Listing 6: Distribución incial no informativa

```
1
2  N <- n
3  modelo_normal_sigma.txt <-
4    '
5  model{
6    for(i in 1:N){
7      x[i] ~ dnorm(0, nu)
8    }
9    # iniciales
10   nu <- 1/sigma2
11   sigma2 ~ dunif(0.1, 300)
12 }
13 '
14
15 cat(modelo_normal_sigma.txt, file = 'modelo_normal_sigma.bugs')
16
17 # valores iniciales para los parametros, si no se especifican la funcion jags
18 # generara valores iniciales
19 jags.inits <- function(){
20   list("nu" = runif(1, 0.1, 300))
21 }
22
23 jags_fit_sigma <- jags(
24   model.file = "modelo_normal_sigma.bugs", # modelo de JAGS
25   #inits = jags.inits, # valores iniciales
26   data = list(x = x, N = N), # lista con los datos
27   parameters.to.save = c("nu", "sigma2"), # parametros por guardar
28   n.chains = 3, # numero de cadenas
29   n.iter = 10000, # numero de pasos
30   n.burnin = 1000, # calentamiento de la cadena
31   n.thin = 1
32 )
33
34 jags_fit_sigma
35
36 # podemos ver las cadenas
37 traceplot(jags_fit_sigma, varname = c("nu", "sigma2"))
38
39 head(jags_fit_sigma$BUGSoutput$summary)
40
41 sigmas <- (jags_fit_sigma$BUGSoutput$sims.matrix[, 3])
42 sigma2_jags <- mean(sigmas)
43 sigma2_jags
```

Y a partir de las simulaciones obtenidas se puede obtener el histograma de la distribución posterior, así como su media que es de $\hat{\sigma^2} = 134.8265$ y su error

estándar $es = 15.85616$.

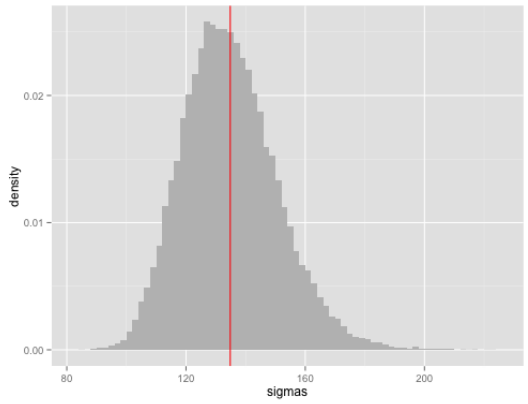

Figure 7: Histograma de simulaciones de la distribución posterior con JAGS

4.2.2 Comparación de los Resultados

A continuación se muestra una tabla comparativa de los tres métodos:

	$\hat{\sigma^2}$	es
Bootstrap	131.0694	15.19693
Bayes	131.1105	15.35968
JAGS	134.8265	15.85616

Donde es posible observar que tanto el enfoque de bootstrap y el bayesiano llegan a soluciones similares, mientras que el enfoque a través de JAGS se queda cerca de los otros dos resultados, esto es debido a que la distribución inicial que se usó en el muestreador de Gibbs fue una distribución no informativa, es decir, esta asume que no tenemos conocimiento previo sobre la distribución de σ^2. Por esta razón, su valor estimado difiere, sin embargo para haber partido de una uniforme como inicial, se encuentra demasiado cerca del estimado por los otros modelos.

4.3 Pregunta 3: Estimación sobre $ln(\sigma)$

Supongamos que ahora buscamos hacer inferencia del parámetro $\tau = ln(\sigma)$ y deseamos usar la técnica de bootstrap paramétrico y la inferencia bayesiana para comparar una vez más sus resultados.

Para el enfoque de bootstrap paramétrico es necesario conocer el estimador de máxima verosimilutd del parámetro para que a partir de dicha estimación se pueda simular realizaciónes de la variable y observar su distribución posterior. Sin embargo, debido a que los estimadores de máxima verosimilitud son equivariantes, el estimador de máxima verosimilitud de τ es $\hat{\tau} = ln(\hat{\sigma})$.

De esta manera es posible obtener el siguiente histograma de las replicaciones bootstrap junto con su intervalo de confianza al 95% que es $(2.32, 2.55)$.

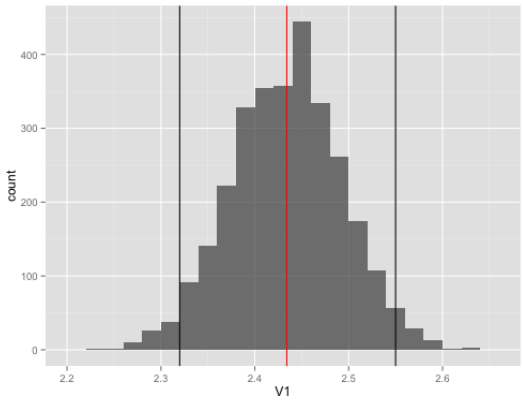

Figure 8: Histograma de las replicaciones bootstrap para $log(\sigma)$

Ahora volvamos a hacer inferencia bayesiana utilizando la inicial uniforme para σ^2. El histograma de la distribución posterior en este caso es el que se muestra a continuación junto con su intervalo de confianza que es $(2.34, 2.57)$.

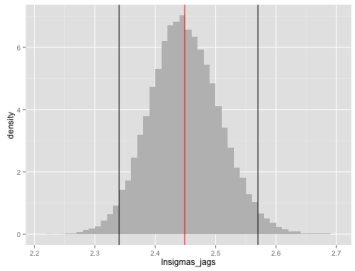

Figure 9: Histograma de las distribución posterior para $log(\sigma)$

El código usado fue:

Listing 7: Análisis $ln(\sigma)$

```
1
2  ################a
3  ## bootstrap
4  ################a
5
6  # estimador de max verosimilitud es
7  lnsigma <- log(sqrt(1 / n * sum((x - 0) ^ 2)))
8
9  # utilizamos bootstrap parametrico
10 lnsigmaBoot <- function(){
11   # Simular X_1*,...X_N* con distribucion N(mu_hat, sigma_hat^2)
12   x_boot <- rnorm(n, mean = 0, sd = sqrt(sigma2))
13   # Calcular mu*, sigma* y theta*
14   lnsigma_boot <- log(sqrt(1 / n * sum((x_boot - 0) ^ 2)))
15   lnsigma_boot
16 }
17
18 # Paso 4: Repetimos B = 3000 veces estimamos el error estandar
19 sims_lnsigma_boot <- rdply(3000, lnsigmaBoot())
20 lnsigma_boot <- mean(sims_lnsigma_boot$V1)
21 lnsigma_boot
22
23 # histograma de las replicaciones bootstrap
24 ggplot(sims_lnsigma_boot,aes(x=V1))+
25   geom_histogram(binwidth = .05)
26
27
28 # como los cuantiles de la distribucion de la estimacion de lnsigma son muy parecidos
29 # a los cuantiles de una normal, usaremos los cuantiles del histograma bootstrap para
30 # definir los intervalos de confianza
31
32 round(quantile(sims_lnsigma_boot$V1, probs = c(0.025, 0.05, 0.1, 0.9, 0.95, 0.975)), 2)
33 round(qnorm(p = c(0.025, 0.05, 0.1, 0.9, 0.95, 0.975), mean = lnsigma,
34             sd = sd(sims_lnsigma_boot$V1)), 2)
35
36 # intervalo de confianza por percentiles
37 ls_per <- round(quantile(sims_lnsigma_boot$V1, prob = 0.975), 2)
38 li_per <- round(quantile(sims_lnsigma_boot$V1, prob = 0.025), 2)
39
40
41 # hisograma con intervalos de confianza
42 ggplot(sims_lnsigma_boot, aes(x = V1)) +
43   geom_histogram(binwidth = 0.02, alpha=.7,fill = "gray30") +
44   geom_vline(xintercept = c(li_per, ls_per, lnsigma_boot),
45              color = c("black", "black", "red"))
46
47
48 ################a
49 ## jags
50 ################a
51
52
53 N <- n
```

```
modelo_normal_tau.txt <-
  '
model{
for(i in 1:N){
x[i] ~ dnorm(0, nu)
}
# iniciales
nu <- 1/sigma2
sigma2 ~ dunif(0.1, 300)
tau <- log(sqrt(sigma2))
}
'

cat(modelo_normal_tau.txt, file = 'modelo_normal_tau.bugs')

# valores iniciales para los parametros, si no se especifican la funcion jags
# generara valores iniciales
#jags.inits <- function(){
# list("nu" = runif(1, 0.1, 300))
#}

jags_fit_tau <- jags(
  model.file = "modelo_normal_tau.bugs", # modelo de JAGS
  #inits = jags.inits, # valores iniciales
  data = list(x = x, N = N), # lista con los datos
  parameters.to.save = c("nu", "sigma2", "tau"), # parametros por guardar
  n.chains = 3, # numero de cadenas
  n.iter = 10000, # numero de pasos
  n.burnin = 1000, # calentamiento de la cadena
  n.thin = 1
)

jags_fit_tau

# podemos ver las cadenas
traceplot(jags_fit_tau, varname = c("nu", "sigma2"))

head(jags_fit_tau$BUGSoutput$summary)

lnsigmas_jags <- (jags_fit_tau$BUGSoutput$sims.matrix[, 4])
lnsigma_jags <- mean(lnsigmas_jags)
lnsigma_jags

# intervalo de confianza por percentiles
ls_per_jags <- round(quantile(lnsigmas_jags, prob = 0.975), 2)
li_per_jags <- round(quantile(lnsigmas_jags, prob = 0.025), 2)



# histograma posterior
ggplot(as.data.frame(lnsigmas_jags), aes(x = lnsigmas_jags)) +
  geom_histogram(aes(y = ..density..), binwidth = .01, fill = "gray") +
  geom_vline(xintercept = c(li_per_jags, ls_per_jags, lnsigma_jags),
             color = c("black", "black", "red"))
```

5 Pregunta 4: Metrópolis

Anteriormente se programó un algoritmo Metropolis para el caso Normal con varianza conocida. En el ejercicio de la tarea los saltos se proponían de acuerdo a una distribución normal: $N(0, 5)$. Para este ejercicio modifica el código con el fin de calcular el porcentaje de valores rechazados y considera las siguientes distribuciones propuesta: a) $N(0, 0.2)$, b) $N(0, 5)$ y c) $N(0, 20)$.

En primer lugar debemos obtener la distribución posterior considerando caa una de las distribuciones propustas y graficar los primeros 2000 pasos de cada cadena. Los resultados fueron los siguientes:

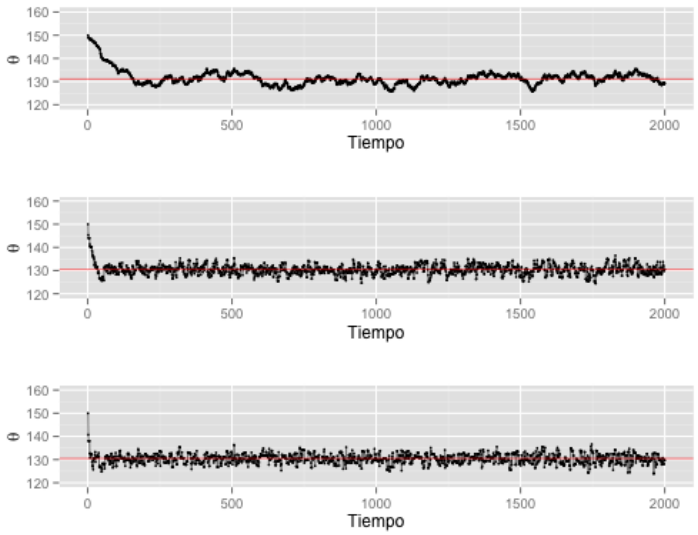

Figure 10: Primeros 2000 pasos para cada cadena

Se puede observar que todas las cadenas convergen al mismo valor, sin embargo, aquella cadena que tiene por distribución propuesta a una $N(0, 0.2)$ debido a la poca varianza en cada uno de los pasos, tarda más en converger al valor real. Por otro lado, las siguientes dos cadenas convergen suficientemente rápido al valor real con la diferencia de que la última de ellas se ve más ruidosa por la varianza tan grande que tiene su distribución propuesta.

El porcentaje de valores rechazados en cada una de las cadenas es el siguiente:

Donde se puede observar que, tal y como se esperaba el proceso que tiene una tasa de rechazo más alta, es aquel que tiene una función de propuesta con mayor varianza, mientras que aquel que tiene menor varianza, aunque tarda

Porcentaje de valores rechazados		
$N(0, 0.2)$	$N(0, 5)$	$N(0, 20)$
0.0785	0.321	0.5348

mas en converger, la tasa de rechazo es relativamente baja.

Si ahora nos interesa ver cuál es la distribución propuesta en cada uno de estos procesos, entonces debemos en primer lugar, deshacernos de la etapa de calentamiento de cada proceso y posteriormente obtener los histogramas de la distribución posterior, en el siguiente gráfico se aprecia que el segundo histograma es el que se asemeja en mayor medida a la distribución posterior verdadera:

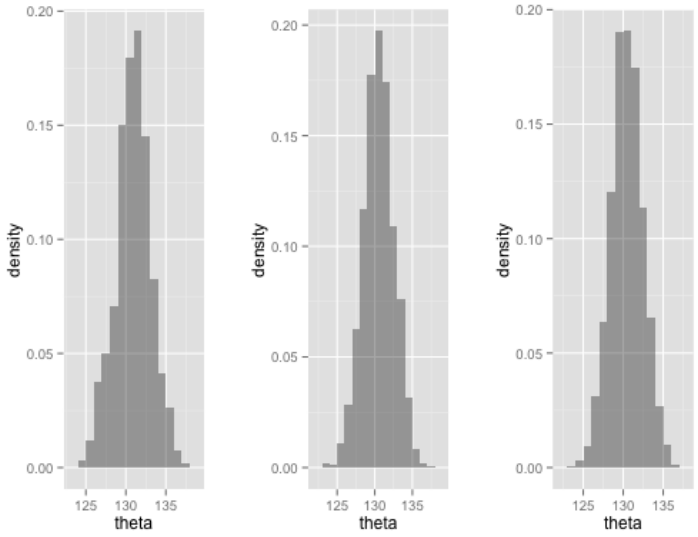

Figure 11: Histogramas de la distribución posterior

Finalmente, es de nuestro interés presentar el histograma de la distribución predictiva posterior, ya que a partir de ella podemos decir cuál podría ser el siguiente valor a observar, así como un intervalo del 95% de probabilidad para la predicción $(126.19, 135)$, tal y como se muestra a continuación:

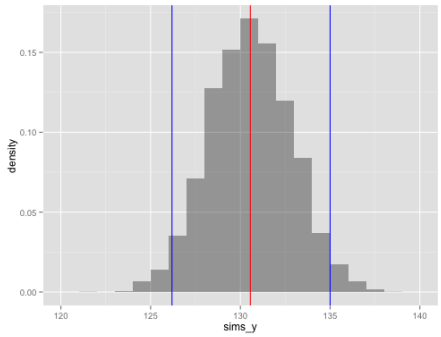

Figure 12: Histogramas de la distribución predictiva posterior

El código que se utilizó a lo largo de esta sección se muestra a continuación:

Listing 8: R example

```
1
2  prior <- function(mu, tau){
3    function(theta) {
4      1/sqrt(2*pi*tau^2)*exp(-1/(2*tau^2)*((theta-mu)^2))
5    }
6  }
7
8  mi_prior<-prior(150,15)
9
10 likeNorm <- function(S, S2, N){
11   function(theta){
12     (1/(800*pi))^(N/2)*exp(-(1/800)*(S2-2*theta*S+N*(theta^2)))
13   }
14 }
15
16 mi_like<-likeNorm(13000,1700000,100)
17
18 postRelProb <- function(theta){
19   mi_like(theta) * mi_prior(theta)
20 }
21
22
23 ### a) rechazados para N(0,0.2)
24 #Metropolis
25
26 set.seed(114041)
27 # para cada paso decidimos el movimiento de acuerdo a la siguiente funcion
28 caminaAleat <- function(theta){ # theta: valor actual
29   salto_prop <- rnorm(1, 0, sd = sqrt(0.2)) # salto propuesto
30   theta_prop <- theta + salto_prop # theta propuesta
31   #if(theta_prop < 0 | theta_prop > 1){ # si el salto implica salir del dominio
32   # return(theta)
33   #}
34   u <- runif(1)
```

```
  p_move = min(postRelProb(theta_prop) / postRelProb(theta), 1) # prob mover
  if(p_move > u){
    return(theta_prop) # aceptar valor propuesto
  }
  else{
    return(theta) # rechazar
  }
}

pasos <- 6000
camino_A <- numeric(pasos) # vector que guardara las simulaciones
camino_A[1] <- 150 # valor inicial

rechazo = 0

# Generamos la caminata aleatoria
for (j in 2:pasos){
  camino_A[j] <- caminaAleat(camino_A[j-1])
  rechazo <- rechazo + 1 * (camino_A[j] == camino_A[j - 1])
}


caminata <- data.frame(pasos = 1:pasos, theta = camino_A)

pasosN02 <- ggplot(caminata[1:2000, ], aes(x = pasos, y = theta)) +
  geom_point(size = 0.8) +
  geom_path(alpha = 0.5) +
  scale_y_continuous(expression(theta), limits = c(120, 160)) +
  scale_x_continuous("Tiempo") +
  geom_hline(yintercept = mean(caminata$theta), color = "red", alpha = 0.5)


rechazo / pasos
### 0.0785



### b) rechazados para N(0,5)
#Metropolis

set.seed(114041)
# para cada paso decidimos el movimiento de acuerdo a la siguiente funcion
caminaAleat <- function(theta){ # theta: valor actual
  salto_prop <- rnorm(1, 0, sd = sqrt(5)) # salto propuesto
  theta_prop <- theta + salto_prop # theta propuesta
  #if(theta_prop < 0 | theta_prop > 1){ # si el salto implica salir del dominio
  # return(theta)
  #}
  u <- runif(1)
  p_move = min(postRelProb(theta_prop) / postRelProb(theta), 1) # prob mover
  if(p_move > u){
    return(theta_prop) # aceptar valor propuesto
  }
  else{
    return(theta) # rechazar
  }
}
```

```r

pasos <- 6000
camino_B <- numeric(pasos) # vector que guardara las simulaciones
camino_B[1] <- 150 # valor inicial

rechazo = 0

# Generamos la caminata aleatoria
for (j in 2:pasos){
  camino_B[j] <- caminaAleat(camino_B[j-1])
  rechazo <- rechazo + 1 * (camino_B[j] == camino_B[j - 1])
}


caminata <- data.frame(pasos = 1:pasos, theta = camino_B)

pasosN05 <- ggplot(caminata[1:2000, ], aes(x = pasos, y = theta)) +
  geom_point(size = 0.8) +
  geom_path(alpha = 0.5) +
  scale_y_continuous(expression(theta), limits = c(120, 160)) +
  scale_x_continuous("Tiempo") +
  geom_hline(yintercept = mean(caminata$theta), color = "red", alpha = 0.5)
pasosN05

rechazo / pasos
### 0.3216667

### c) rechazados para N(0,20)
#Metropolis

set.seed(114041)
# para cada paso decidimos el movimiento de acuerdo a la siguiente funcion
caminaAleat <- function(theta){ # theta: valor actual
  salto_prop <- rnorm(1, 0, sd = sqrt(20)) # salto propuesto
  theta_prop <- theta + salto_prop # theta propuesta
  #if(theta_prop < 0 | theta_prop > 1){ # si el salto implica salir del dominio
  # return(theta)
  #}
  u <- runif(1)
  p_move = min(postRelProb(theta_prop) / postRelProb(theta), 1) # prob mover
  if(p_move > u){
    return(theta_prop) # aceptar valor propuesto
  }
  else{
    return(theta) # rechazar
  }
}

pasos <- 6000
camino_C <- numeric(pasos) # vector que guardara las simulaciones
camino_C[1] <- 150 # valor inicial

rechazo = 0

# Generamos la caminata aleatoria
for (j in 2:pasos){
  camino_C[j] <- caminaAleat(camino_C[j-1])
```

```
  rechazo <- rechazo + 1 * (camino_C[j] == camino_C[j - 1])
}

caminata <- data.frame(pasos = 1:pasos, theta = camino_C)

pasosN020 <- ggplot(caminata[1:2000, ], aes(x = pasos, y = theta)) +
  geom_point(size = 0.8) +
  geom_path(alpha = 0.5) +
  scale_y_continuous(expression(theta), limits = c(120, 160)) +
  scale_x_continuous("Tiempo") +
  geom_hline(yintercept = mean(caminata$theta), color = "red", alpha = 0.5)
pasosN020

rechazo / pasos
### 0.5348333


### a) valores de la distribucion posterior y grafica para N(0,0.2)

caminata_A <- data.frame(pasos = 1:pasos, theta = camino_A)
grafica_A <- ggplot(caminata_A[1:2000, ], aes(x = pasos, y = theta)) +
                    geom_point(size = 0.8) +
                    geom_path(alpha = 0.3)
grafica_A

### b) valores de la distribucion posterior y grafica para N(0,5)

caminata_B <- data.frame(pasos = 1:pasos, theta = camino_B)
grafica_B <- ggplot(caminata_B[1:2000, ], aes(x = pasos, y = theta)) +
  geom_point(size = 0.8) +
  geom_path(alpha = 0.3)
grafica_B


### c) valores de la distribucion posterior y grafica para N(0,20)

caminata_C <- data.frame(pasos = 1:pasos, theta = camino_C)
grafica_C <- ggplot(caminata_C[1:2000, ], aes(x = pasos, y = theta)) +
  geom_point(size = 0.8) +
  geom_path(alpha = 0.3)
grafica_C


grid.arrange(grafica_A, grafica_B, grafica_C, ncol = 1)
grid.arrange(pasosN02, pasosN05, pasosN020, ncol = 1)

### a) histograma de la distribucion posterior para N(0,0.2)
hist_A <- ggplot(filter(caminata_A, pasos > 1000), aes(x = theta)) +
                  geom_histogram(aes(y = ..density..),binwidth=1,alpha=.4)
hist_A

### b) histograma de la distribucion posterior para N(0,5)
hist_B <- ggplot(filter(caminata_B, pasos > 1000), aes(x = theta)) +
                   geom_histogram(aes(y = ..density..),binwidth=1,alpha=.4)
hist_B

### c) histograma de la distribucion posterior para N(0,20)
```

```r
hist_C <- ggplot(filter(caminata_C, pasos > 1000), aes(x = theta)) +
                    geom_histogram(aes(y = ..density..),binwidth=1,alpha=.4)
hist_C

grid.arrange(hist_A, hist_B, hist_C, ncol = 3)


### histograma de la probabilidad predictiva posterior
caminata_f <- filter(caminata_B, pasos > 1000)

sims_y <- rnorm(nrow(caminata_f), mean = mean(caminata_f$theta), sd=sqrt(5) )
mean(sims_y) # p(y = 1 | x) probabilidad predictiva

ls_per_pred <- round(quantile(as.data.frame(sims_y)$sims_y, prob = 0.975), 2)
li_per_pred <- round(quantile(as.data.frame(sims_y)$sims_y, prob = 0.025), 2)

ggplot(as.data.frame(sims_y), aes(x = sims_y)) +
  geom_histogram(aes(y = ..density..),binwidth=1,alpha=.4) +
  geom_vline(xintercept = c(mean(sims_y),li_per_pred,ls_per_pred),
             color = c("red","blue","blue"))
```

6 Pregunta 5: Convergencia

Implementaremos un modelo de regresión en JAGS, la base de datos que usaremos contiene información de mediciones de radón (activity) y del suelo en el que se hicieron las mediciones (floor = 0 casas con sótano, floor = 1 casas sin sótano), las mediciones corresponden a 919 hogares muestreados de 85 condados de Minnesota. El objetivo es construir un modelo de regresión en el que la medición de radón es la variable dependiente y el tipo de suelo es la covariable.

El modelo es como sigue:

$$y_i \sim N(\alpha + \beta x_i, \sigma^2) \tag{13}$$

La distribuciones iniciales que usaremos son:

$$\begin{aligned} \beta &\sim N(0, 1000) \\ \sigma^2 &\sim U(0, 1000) \end{aligned}$$

Listing 9: Modelo Regresión

```r

library(R2jags)

modelo_regresion.txt <-
'
model{
```

```
  for(i in 1 : n) {
    y[i] ~ dnorm(y.hat[i], tau.y)
    y.hat[i] <- a + b * x[i]
  }
  a ~ dnorm(0, 0.001)
  b ~ dnorm(0, 0.001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif(0, 100)
}
'
cat(modelo_regresion.txt, file = 'modelo_regresion.bugs')

### Radon
load("data/radon.Rdata")

# Iniciamos preparando los datos para el analisis, trabajaremos en
# escala logaritmica, hay algunos casos con medicion cero, para estos
# hacemos una pequena correcion redondeandolos a 0.1.
y <- log(ifelse (radon.2$activity == 0, 0.1, radon.2$activity))
n <- nrow(radon.2)
x <- radon.2$floor

# jags
radon1.data <- list("n", "y", "x")
radon1.inits <- function(){
  list(a = rnorm(1),
  b = rnorm(1),
  sigma.y = runif(1))}

radon1.parameters <- c("a", "b", "sigma.y")
```

El ejercicio consiste en que definas número de cadenas, número de iteraciones y etapa de calentamiento en la siguiente instrucción. Asegurate de alcanzar convergencia y describe los diagnósticos que utilizaste para concluir que se convergió a la distribución posterior.

Listing 10: Definición cadenas iteraciones y calentamiento

```

radon1.jags <- jags(
  model.file = "modelo_regresion.bugs",
  inits= radon1.inits,
  data = radon1.data,
  parameters.to.save = radon1.parameters,
  n.chains = 3, # numero de cadenas
  n.iter = 10000, # numero de pasos
  n.burnin = 1000, # calentamiento de la cadena
  n.thin = 1)
```

Para asegurarnos de haber alcanzado la convergencia existen diversos diagnósticos que podemos seguir para concluir si se ha logrado la convergencia o no. Dichos diagnósticos están muy relacionados con la definición de ciertos elementos a la hora de ajustar el modelo, es decir, se propone trabajar con más

de una cadena ya que así podemos ver si realmente el proceso se ha olvidado del estado inicial viendo si las simulaciones convergen a una distribución en común. El factor de reducción potencial de escala nos ayuda a determinar una posible reducción en la longitud de un intervalo de confianza si las simulaciones continuaran infinitamente, en este estadístico nos gustaría que en general no se encuentre muy por encima del 1, sin embargo, hay opciones en la iteración para lograr que cada parámetro alcancen un valor de 1.1. Finalmente, el número efectivo de simulaciones nos ayuda a identificar como su nombre lo indica el número efectivo de simulaciones dado que las cadenas podrían llegar a estar muy correlacionadas.

En la siguiente tabla y gráficos podemos observar las caractrísticas arriba mencionadas:

```
Inference for Bugs model at "modelo_regresion.bugs", fit using jags,
 3 chains, each with 10000 iterations (first 1000 discarded)
 n.sims = 27000 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat n.eff
a           1.327   0.030    1.268    1.307    1.327    1.347    1.385 1.001 27000
b          -0.614   0.073   -0.756   -0.663   -0.614   -0.564   -0.468 1.001 21000
sigma.y     0.824   0.019    0.787    0.810    0.823    0.836    0.863 1.001 27000
deviance 2250.056   2.465 2247.242 2248.243 2249.429 2251.181 2256.393 1.001 27000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 3.0 and DIC = 2253.1
DIC is an estimate of expected predictive error (lower deviance is better).
```

Figure 13: Diagnósticos

Es posible ver que tanto el factor de reducción potencial de escala y el número efectivo de simulaciónes cumplen con los requisitos necesarios para poder determinar convergencia. Solamente hace falta observar si las tres cadenas que se comportan de manera similar.

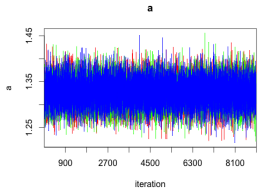

Figure 14: Parámetro a

Y como era de esperarse, todas las cadenas tienen un comportamiento muy similar. Por lo que en conclusión de acuerdo a los diagnósticos propuestos, se puede asegurar que se logró con éxito la convergencia.

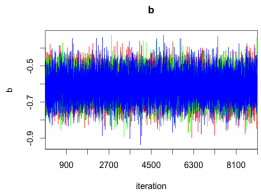

Figure 15: Parámetro b

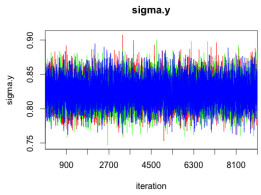

Figure 16: Parámetro σ^2

7 Pregunta 6: Modelos Jerárquicos

En este ejercicio definirás un modelo jerárquico para la incidencia de tumores en grupos de conejos a los que se suministró una medicina. Se realizaron 71 experimentos distintos utilizando la misma medicina.

Considerando que cada conejo proviene de un experimento distinto, se desea estudiar θ_j, la probabilidad de desarrollar un tumor en el jésimo grupo, este parámetro variará de grupo a grupo.

Denotaremos y_{ij} la observación en el i-ésimo conejo perteneciente al j-ésimo experimento, y_{ij} puede tomar dos valores: 1 indicando que el conejo desarrolló tumor y 0 en el caso contrario.

$$y_{ij} \sim Bernoulli(\theta_j) \tag{14}$$

Adicionalmente se desea estimar el efecto medio de la medicina a lo largo de los grupos μ, por lo que utilizaremos un modelo jerárquico como sigue:

$$\theta_j \sim Beta(a, b) \tag{15}$$

donde

$$\begin{aligned} a &= \mu\kappa \\ b &= (1-\mu)\kappa \end{aligned}$$

Finalmente asignamos una distribución a los hiperparámetros μ y κ,

$$\begin{aligned} \mu &\sim Beta(A_\mu, B_\mu) \\ \kappa &\sim Gamma(S_\kappa, R_\kappa) \end{aligned}$$

Si pensamos en este problema como un problema de lanzamiento de monedas, la variable y_{ij} (la obervación del i-ésimo conejo perteneciente al j-ésimo experimento, corresponde a la realización de observar águila o solo al lanzar la moneda y la variable θ_j que es la probabilidad de desarrollar un tumor en el j-ésimo grupo, corresponde en el caso de las monedas al sesgo, además se tienen tantas θ_j como monedas que participan en el experimento.

La base de datos rabbits contiene las observaciones de los 71 experimentos, cada renglón corresponde a una observación.

Utiliza JAGS para ajustar un modelo jerárquico como el descrito arriba y usando una inicial $Beta(1, 1)$ y una $Gamma(1, 0.1)$ para μ y κ respectivamente.

El código que muestra como se realiza el modelo jerárquico con estas distribuciones iniciales es:

Listing 11: Modelo Jerárquico

```

load("rabbits.RData")
head(rabbits)
# hay 21 realizaciones en cada experimento

modelo_jer.txt <-
  '
model{
for(t in 1:N){
x[t] ~ dbern(theta[grupo[t]])
}
for(j in 1:nGrupos){
theta[j] ~ dbeta(a, b)
}
a <- mu * kappa
b <- (1 - mu) * kappa
mu ~ dbeta(1, 1)
kappa ~ dgamma(1, 0.1)
}
'

cat(modelo_jer.txt, file = 'modelo_jer.bugs')

jags_fit_rab <- jags(
  model.file = "modelo_jer.bugs", # modelo de JAGS
  #inits = jags.inits, # valores iniciales
  data = list(x = rabbits$tumor, grupo = rabbits$experiment, nGrupos = 71, N = nrow(rabbits)),
# lista con los datos
  parameters.to.save = c("mu", "kappa", "theta"), # parametros por guardar
  n.chains = 3, # numero de cadenas
```

```
  n.iter = 10000, # numero de pasos
  n.burnin = 1000 # calentamiento de la cadena
)

traceplot(jags_fit_rab, varname = c("kappa", "mu"))

# validacion de convergencia
jags_fit_rab


# histograma de la distribucion posterior para mu y para kappa

sims_df <- data.frame(n_sim = 1:jags_fit_rab$BUGSoutput$n.sims,
                      jags_fit_rab$BUGSoutput$sims.matrix) %>%
  dplyr::select(-deviance) %>%
  gather(parametro, value, -n_sim)

mu_rab_62 <- ggplot(filter(sims_df, parametro == "mu"), aes(x = value)) +
  geom_histogram(alpha = 0.8, binwidth=.003)

kappa_rab_62 <- ggplot(filter(sims_df, parametro == "kappa"), aes(x = value)) +
  geom_histogram(alpha = 0.8, binwidth=4)

grid.arrange(mu_rab_62, kappa_rab_62, ncol = 2)
```

Una vez ajustado el modelo es posible realizar histogramas de la distribución posterior de μ y de κ, dichos histogramas se muestran a continuación:

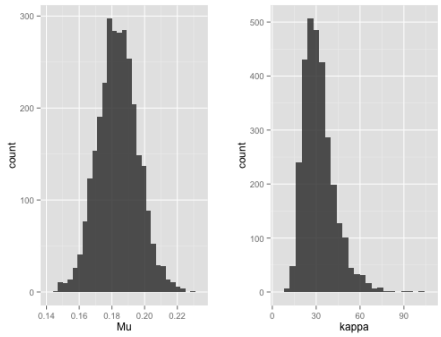

Figure 17: Histogramas de la distribución posterior de μ y κ

Observando los histogramas es posible notar que la distribución de κ está sesgada a la derecha; debido a que el parámetro κ mide el grado de dependencia entre μ y θ_j entonces es posible asegurar que los valores de θ_j se encuentran al rededor de μ con cierta variación, mientras que si la distribución huiera estado sesgada a la izquierda entonces se observaría que los valores de θ_j estarían muy cercanos de μ.

Si por otro lado, la distribución posterior de μ describe nuestra incertidumbre acerca de si un conejo puede desarrollar un tumor o no y de acuerdo al histograma se puede suponer que la propensión es baja.

Realizamos una gráfica de boxplots con las simulaciones de cada parámetro θ_j.

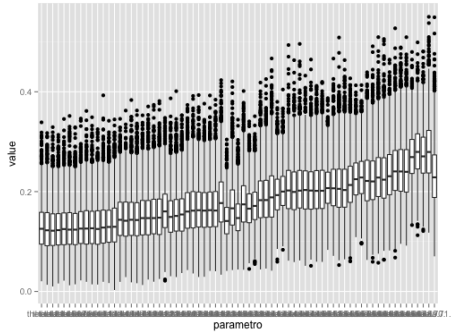

Figure 18: Boxplots de las simulaciones de cada parámetro θ_j

De acuerdo a las boxplot para cada uno de los parámetros θ_j es posible observar que conforme la muestra de conejos aumenta, la propensión a observar que el conejo desarrolla un tumor también aumenta.

Finalmente, ajustamos un nuevo modelo utilizando una incial $Beta(10, 10)$ y una $Gamma(0.51, 0.01)$ para μ y κ y realizamos una gráfica que nos permite comparar las medias posteriores de los parámetros θ_j bajo ambos escenarios de distribuciones iniciales.

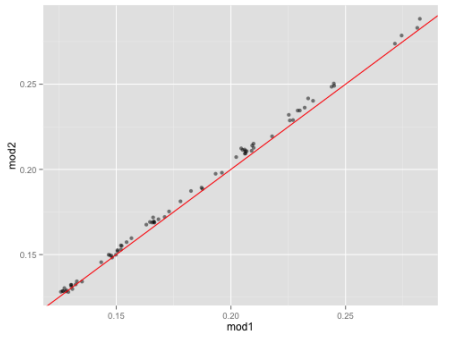

Figure 19: Comparación de medias posterirores de los modelos

En la gráfica anterior se puede observar que el modelo 2, es decir aquel que

tiene por distribuciones iniciales una $Beta(10, 10)$ y una $Gamma(0.51, 0.01)$ para μ y κ tiende a sobreestimar el valor de θ_j pues como se puede apreciar, se encuentran por encima de la identidad $x = y$ la mayor parte del tiempo.

El código usado en esta última sección es:

Listing 12: Comparación de medias posteriores de modelos

```
1
2
3  ################a
4  ## 6.3
5  ################a
6
7  ggplot(subset(sims_df, parametro != "kappa" & parametro != "mu"), aes(x = parametro, y = value)) +
8    geom_boxplot()
9
10 ################a
11 ## 6.4
12 ################a
13
14 modelo_jer_64.txt <-
15   '
16 model{
17 for(t in 1:N){
18 x[t] ~ dbern(theta[grupo[t]])
19 }
20 for(j in 1:nGrupos){
21 theta[j] ~ dbeta(a, b)
22 }
23 a <- mu * kappa
24 b <- (1 - mu) * kappa
25 mu ~ dbeta(10, 10)
26 kappa ~ dgamma(.51, 0.1)
27 }
28 '
29
30 cat(modelo_jer_64.txt, file = 'modelo_jer_64.bugs')
31
32 jags_fit_rab_64 <- jags(
33   model.file = "modelo_jer_64.bugs", # modelo de JAGS
34   #inits = jags.inits, # valores iniciales
35   data = list(x = rabbits$tumor, grupo = rabbits$experiment, nGrupos = 71, N = nrow(rabbits)),
   # lista con los datos
36   parameters.to.save = c("mu", "kappa", "theta"), # parametros por guardar
37   n.chains = 3, # numero de cadenas
38   n.iter = 10000, # numero de pasos
39   n.burnin = 1000 # calentamiento de la cadena
40 )
41
42 traceplot(jags_fit_rab_64, varname = c("kappa", "mu"))
43
44 # validacion de convergencia
45 head(jags_fit_rab_64)
46
47
48 #grafica comoparacion medias posteriores
49 mod1 <- as.data.frame(jags_fit_rab$BUGSoutput$summary)
50 mod2 <- as.data.frame(jags_fit_rab_64$BUGSoutput$summary)
51 mod1 <- mod1[-c(1,2,3),1]
52 mod2 <- mod2[-c(1,2,3),1]
```

```
53 thetas_modelos <- as.data.frame(cbind(mod1,mod2))
54
55
56 ggplot(thetas_modelos, aes(x=mod1, y=mod2)) +
57   geom_point(alpha=.5) +
58   geom_abline(intercept = 0, colour="red")
```