Tarea 6

Juan M Karawcki, Bruno Pintos

2025-09-22

Ejercicio 6.7 (Aproximación por grilla Normal-Normal)

Consideremos el modelo Normal–Normal para μ con

$$Y_i \mid \mu \sim N(\mu,\, 1.3^2)$$

y

$$\mu \sim N(10,\, 1.2^2)$$

Supongamos que en $n = 4$ observaciones independientes se obtienen los datos:

$$(Y_1, Y_2, Y_3, Y_4) = (7.1,\, 8.9,\, 8.4,\, 8.6)$$

1. Utiliza la aproximación por grilla con valores de

$$\mu \in \{5, 6, 7, \ldots, 15\}$$

para aproximar el modelo posterior de μ.

```
# PASO 1: Definir una grilla de 11 valores de mu
grid_data <- data.frame(mu_grid = seq(from = 5, to = 15, by = 1))
y <- c(7.1, 8.9, 8.4, 8.6)
```

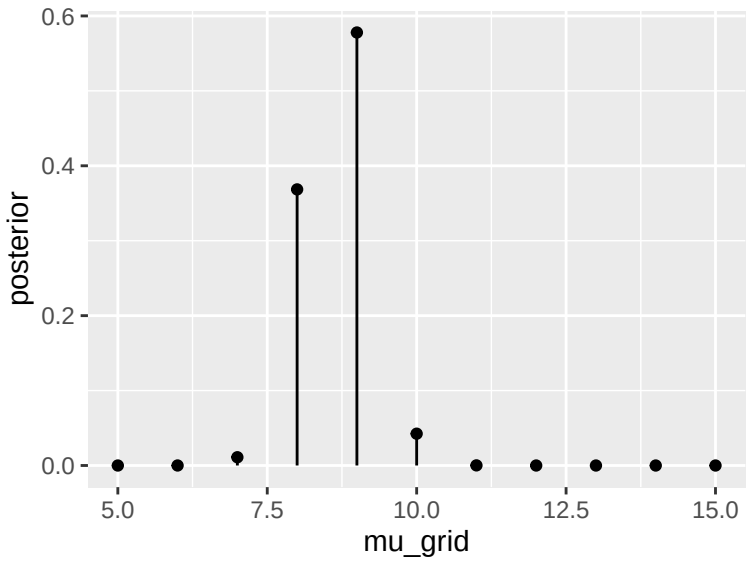
```
# PASO 2: Evaluar la priori y la verosimilitud para cada valor de mu

grid_data <- grid_data %>%
  mutate(
    prior = dnorm(mu_grid, 10, 1.2),
    # Priori: distribución normal(10,1.2**2)
    likelihood = sapply(mu_grid, function(m) prod(dnorm(y, mean = m, sd = 1.3)))
    # Verosimilitud:
  )
```

```
# PASO 3: Aproximación de la distribución a posteriori (no normalizada y normalizada)
grid_data <- grid_data %>%
  mutate(
    unnormalized = likelihood * prior,
```

```
    # Producto prior * verosimilitud
    posterior = unnormalized / sum(unnormalized)
    # Normalización para que sume 1
  )
```

```
# Verificar que la suma de las probabilidades que
# definen la aproximación computacional de la
# distribución a posteriori normalizada es 1
grid_data %>%
  summarize(sum(unnormalized), sum(posterior))
```

```
##   sum(unnormalized) sum(posterior)
## 1        0.00105926              1
```



2. Repite la parte (a) usando una grilla de 201 valores igualmente espaciados entre 5 y 15.

```
# PASO 1: Definir una grilla de 11 valores de mu
grid_data <- data.frame(mu_grid = seq(from = 5, to = 15, length.out = 201))
y <- c(7.1, 8.9, 8.4, 8.6)
```

```
# PASO 2: Evaluar la priori y la verosimilitud para cada valor de mu

grid_data <- grid_data %>%
  mutate(
    prior = dnorm(mu_grid, 10, 1.2),
    # Priori: distribución normal(10,1.2**2)
    likelihood = sapply(mu_grid, function(m) prod(dnorm(y, mean = m, sd = 1.3)))
    # Verosimilitud:
  )
```

```
# PASO 3: Aproximación de la distribución a posteriori (no normalizada y normalizada)
grid_data <- grid_data %>%
```

```
  mutate(
    unnormalized = likelihood * prior,
    # Producto prior * verosimilitud
    posterior = unnormalized / sum(unnormalized)
    # Normalización para que sume 1
  )
```

```
# Verificar que la suma de las probabilidades que
# definen la aproximación computacional de la
# distribución a posteriori normalizada es 1
grid_data %>%
  summarize(sum(unnormalized), sum(posterior))
```

```
##   sum(unnormalized) sum(posterior)
## 1        0.02122574              1
```

```
# Examinar la tabla resultante con la aproximación por grilla
round(grid_data, 5)
```

```
##     mu_grid   prior likelihood unnormalized posterior
## 1      5.00 0.00006    0.00000      0.00000   0.00000
## 2      5.05 0.00007    0.00000      0.00000   0.00000
## 3      5.10 0.00008    0.00000      0.00000   0.00000
## 4      5.15 0.00009    0.00000      0.00000   0.00000
## 5      5.20 0.00011    0.00000      0.00000   0.00000
## 6      5.25 0.00013    0.00000      0.00000   0.00000
## 7      5.30 0.00016    0.00000      0.00000   0.00000
## 8      5.35 0.00018    0.00000      0.00000   0.00000
## 9      5.40 0.00021    0.00000      0.00000   0.00000
## 10     5.45 0.00025    0.00000      0.00000   0.00000
## 11     5.50 0.00029    0.00000      0.00000   0.00000
## 12     5.55 0.00034    0.00000      0.00000   0.00000
## 13     5.60 0.00040    0.00000      0.00000   0.00000
## 14     5.65 0.00047    0.00000      0.00000   0.00000
## 15     5.70 0.00054    0.00000      0.00000   0.00000
## 16     5.75 0.00063    0.00000      0.00000   0.00000
## 17     5.80 0.00073    0.00000      0.00000   0.00000
## 18     5.85 0.00084    0.00001      0.00000   0.00000
## 19     5.90 0.00097    0.00001      0.00000   0.00000
## 20     5.95 0.00112    0.00001      0.00000   0.00000
## 21     6.00 0.00129    0.00001      0.00000   0.00000
## 22     6.05 0.00148    0.00002      0.00000   0.00000
## 23     6.10 0.00169    0.00002      0.00000   0.00000
## 24     6.15 0.00193    0.00003      0.00000   0.00000
## 25     6.20 0.00221    0.00004      0.00000   0.00000
## 26     6.25 0.00252    0.00004      0.00000   0.00001
## 27     6.30 0.00287    0.00006      0.00000   0.00001
## 28     6.35 0.00326    0.00007      0.00000   0.00001
## 29     6.40 0.00369    0.00009      0.00000   0.00002
## 30     6.45 0.00418    0.00011      0.00000   0.00002
## 31     6.50 0.00473    0.00014      0.00000   0.00003
## 32     6.55 0.00533    0.00017      0.00000   0.00004
```

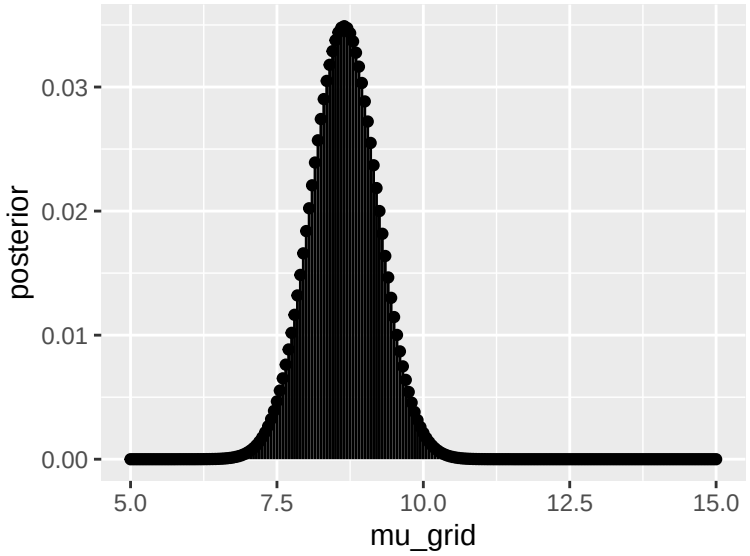
```
## 33     6.60 0.00601    0.00020      0.00000   0.00006
## 34     6.65 0.00675    0.00025      0.00000   0.00008
## 35     6.70 0.00758    0.00030      0.00000   0.00011
## 36     6.75 0.00849    0.00035      0.00000   0.00014
## 37     6.80 0.00950    0.00042      0.00000   0.00019
## 38     6.85 0.01060    0.00050      0.00001   0.00025
## 39     6.90 0.01182    0.00059      0.00001   0.00033
## 40     6.95 0.01315    0.00069      0.00001   0.00043
## 41     7.00 0.01461    0.00080      0.00001   0.00055
## 42     7.05 0.01620    0.00092      0.00001   0.00070
## 43     7.10 0.01793    0.00106      0.00002   0.00090
## 44     7.15 0.01981    0.00121      0.00002   0.00113
## 45     7.20 0.02185    0.00138      0.00003   0.00142
## 46     7.25 0.02406    0.00155      0.00004   0.00176
## 47     7.30 0.02645    0.00174      0.00005   0.00217
## 48     7.35 0.02902    0.00194      0.00006   0.00266
## 49     7.40 0.03179    0.00216      0.00007   0.00323
## 50     7.45 0.03477    0.00238      0.00008   0.00389
## 51     7.50 0.03795    0.00261      0.00010   0.00466
## 52     7.55 0.04136    0.00284      0.00012   0.00553
## 53     7.60 0.04499    0.00308      0.00014   0.00652
## 54     7.65 0.04886    0.00331      0.00016   0.00762
## 55     7.70 0.05297    0.00354      0.00019   0.00885
## 56     7.75 0.05732    0.00377      0.00022   0.01019
## 57     7.80 0.06193    0.00399      0.00025   0.01164
## 58     7.85 0.06678    0.00420      0.00028   0.01320
## 59     7.90 0.07190    0.00439      0.00032   0.01486
## 60     7.95 0.07727    0.00456      0.00035   0.01659
## 61     8.00 0.08290    0.00471      0.00039   0.01839
## 62     8.05 0.08878    0.00484      0.00043   0.02023
## 63     8.10 0.09492    0.00494      0.00047   0.02208
## 64     8.15 0.10130    0.00501      0.00051   0.02391
## 65     8.20 0.10793    0.00506      0.00055   0.02571
## 66     8.25 0.11479    0.00507      0.00058   0.02742
## 67     8.30 0.12188    0.00506      0.00062   0.02903
## 68     8.35 0.12918    0.00501      0.00065   0.03049
## 69     8.40 0.13668    0.00494      0.00067   0.03179
## 70     8.45 0.14436    0.00484      0.00070   0.03289
## 71     8.50 0.15221    0.00471      0.00072   0.03377
## 72     8.55 0.16021    0.00456      0.00073   0.03440
## 73     8.60 0.16833    0.00439      0.00074   0.03478
## 74     8.65 0.17656    0.00420      0.00074   0.03490
## 75     8.70 0.18488    0.00399      0.00074   0.03475
## 76     8.75 0.19325    0.00377      0.00073   0.03434
## 77     8.80 0.20164    0.00354      0.00071   0.03367
## 78     8.85 0.21004    0.00331      0.00070   0.03277
## 79     8.90 0.21841    0.00308      0.00067   0.03164
## 80     8.95 0.22671    0.00284      0.00064   0.03032
## 81     9.00 0.23493    0.00261      0.00061   0.02884
## 82     9.05 0.24302    0.00238      0.00058   0.02722
## 83     9.10 0.25095    0.00216      0.00054   0.02549
## 84     9.15 0.25869    0.00194      0.00050   0.02369
## 85     9.20 0.26621    0.00174      0.00046   0.02185
## 86     9.25 0.27347    0.00155      0.00042   0.02000
```

```
## 87     9.30 0.28044    0.00138      0.00039   0.01817
## 88     9.35 0.28709    0.00121      0.00035   0.01638
## 89     9.40 0.29339    0.00106      0.00031   0.01465
## 90     9.45 0.29930    0.00092      0.00028   0.01301
## 91     9.50 0.30481    0.00080      0.00024   0.01146
## 92     9.55 0.30988    0.00069      0.00021   0.01002
## 93     9.60 0.31449    0.00059      0.00018   0.00869
## 94     9.65 0.31861    0.00050      0.00016   0.00748
## 95     9.70 0.32222    0.00042      0.00014   0.00639
## 96     9.75 0.32531    0.00035      0.00012   0.00542
## 97     9.80 0.32787    0.00030      0.00010   0.00456
## 98     9.85 0.32986    0.00025      0.00008   0.00381
## 99     9.90 0.33130    0.00020      0.00007   0.00316
## 100    9.95 0.33216    0.00017      0.00006   0.00260
## 101   10.00 0.33245    0.00014      0.00004   0.00212
## 102   10.05 0.33216    0.00011      0.00004   0.00172
## 103   10.10 0.33130    0.00009      0.00003   0.00138
## 104   10.15 0.32986    0.00007      0.00002   0.00110
## 105   10.20 0.32787    0.00006      0.00002   0.00087
## 106   10.25 0.32531    0.00004      0.00001   0.00068
## 107   10.30 0.32222    0.00004      0.00001   0.00053
## 108   10.35 0.31861    0.00003      0.00001   0.00041
## 109   10.40 0.31449    0.00002      0.00001   0.00032
## 110   10.45 0.30988    0.00002      0.00001   0.00024
## 111   10.50 0.30481    0.00001      0.00000   0.00018
## 112   10.55 0.29930    0.00001      0.00000   0.00014
## 113   10.60 0.29339    0.00001      0.00000   0.00010
## 114   10.65 0.28709    0.00001      0.00000   0.00008
## 115   10.70 0.28044    0.00000      0.00000   0.00006
## 116   10.75 0.27347    0.00000      0.00000   0.00004
## 117   10.80 0.26621    0.00000      0.00000   0.00003
## 118   10.85 0.25869    0.00000      0.00000   0.00002
## 119   10.90 0.25095    0.00000      0.00000   0.00001
## 120   10.95 0.24302    0.00000      0.00000   0.00001
## 121   11.00 0.23493    0.00000      0.00000   0.00001
## 122   11.05 0.22671    0.00000      0.00000   0.00001
## 123   11.10 0.21841    0.00000      0.00000   0.00000
## 124   11.15 0.21004    0.00000      0.00000   0.00000
## 125   11.20 0.20164    0.00000      0.00000   0.00000
## 126   11.25 0.19325    0.00000      0.00000   0.00000
## 127   11.30 0.18488    0.00000      0.00000   0.00000
## 128   11.35 0.17656    0.00000      0.00000   0.00000
## 129   11.40 0.16833    0.00000      0.00000   0.00000
## 130   11.45 0.16021    0.00000      0.00000   0.00000
## 131   11.50 0.15221    0.00000      0.00000   0.00000
## 132   11.55 0.14436    0.00000      0.00000   0.00000
## 133   11.60 0.13668    0.00000      0.00000   0.00000
## 134   11.65 0.12918    0.00000      0.00000   0.00000
## 135   11.70 0.12188    0.00000      0.00000   0.00000
## 136   11.75 0.11479    0.00000      0.00000   0.00000
## 137   11.80 0.10793    0.00000      0.00000   0.00000
## 138   11.85 0.10130    0.00000      0.00000   0.00000
## 139   11.90 0.09492    0.00000      0.00000   0.00000
## 140   11.95 0.08878    0.00000      0.00000   0.00000
```

```
## 141   12.00 0.08290    0.00000      0.00000   0.00000
## 142   12.05 0.07727    0.00000      0.00000   0.00000
## 143   12.10 0.07190    0.00000      0.00000   0.00000
## 144   12.15 0.06678    0.00000      0.00000   0.00000
## 145   12.20 0.06193    0.00000      0.00000   0.00000
## 146   12.25 0.05732    0.00000      0.00000   0.00000
## 147   12.30 0.05297    0.00000      0.00000   0.00000
## 148   12.35 0.04886    0.00000      0.00000   0.00000
## 149   12.40 0.04499    0.00000      0.00000   0.00000
## 150   12.45 0.04136    0.00000      0.00000   0.00000
## 151   12.50 0.03795    0.00000      0.00000   0.00000
## 152   12.55 0.03477    0.00000      0.00000   0.00000
## 153   12.60 0.03179    0.00000      0.00000   0.00000
## 154   12.65 0.02902    0.00000      0.00000   0.00000
## 155   12.70 0.02645    0.00000      0.00000   0.00000
## 156   12.75 0.02406    0.00000      0.00000   0.00000
## 157   12.80 0.02185    0.00000      0.00000   0.00000
## 158   12.85 0.01981    0.00000      0.00000   0.00000
## 159   12.90 0.01793    0.00000      0.00000   0.00000
## 160   12.95 0.01620    0.00000      0.00000   0.00000
## 161   13.00 0.01461    0.00000      0.00000   0.00000
## 162   13.05 0.01315    0.00000      0.00000   0.00000
## 163   13.10 0.01182    0.00000      0.00000   0.00000
## 164   13.15 0.01060    0.00000      0.00000   0.00000
## 165   13.20 0.00950    0.00000      0.00000   0.00000
## 166   13.25 0.00849    0.00000      0.00000   0.00000
## 167   13.30 0.00758    0.00000      0.00000   0.00000
## 168   13.35 0.00675    0.00000      0.00000   0.00000
## 169   13.40 0.00601    0.00000      0.00000   0.00000
## 170   13.45 0.00533    0.00000      0.00000   0.00000
## 171   13.50 0.00473    0.00000      0.00000   0.00000
## 172   13.55 0.00418    0.00000      0.00000   0.00000
## 173   13.60 0.00369    0.00000      0.00000   0.00000
## 174   13.65 0.00326    0.00000      0.00000   0.00000
## 175   13.70 0.00287    0.00000      0.00000   0.00000
## 176   13.75 0.00252    0.00000      0.00000   0.00000
## 177   13.80 0.00221    0.00000      0.00000   0.00000
## 178   13.85 0.00193    0.00000      0.00000   0.00000
## 179   13.90 0.00169    0.00000      0.00000   0.00000
## 180   13.95 0.00148    0.00000      0.00000   0.00000
## 181   14.00 0.00129    0.00000      0.00000   0.00000
## 182   14.05 0.00112    0.00000      0.00000   0.00000
## 183   14.10 0.00097    0.00000      0.00000   0.00000
## 184   14.15 0.00084    0.00000      0.00000   0.00000
## 185   14.20 0.00073    0.00000      0.00000   0.00000
## 186   14.25 0.00063    0.00000      0.00000   0.00000
## 187   14.30 0.00054    0.00000      0.00000   0.00000
## 188   14.35 0.00047    0.00000      0.00000   0.00000
## 189   14.40 0.00040    0.00000      0.00000   0.00000
## 190   14.45 0.00034    0.00000      0.00000   0.00000
## 191   14.50 0.00029    0.00000      0.00000   0.00000
## 192   14.55 0.00025    0.00000      0.00000   0.00000
## 193   14.60 0.00021    0.00000      0.00000   0.00000
## 194   14.65 0.00018    0.00000      0.00000   0.00000
```

```
## 195   14.70 0.00016    0.00000      0.00000   0.00000
## 196   14.75 0.00013    0.00000      0.00000   0.00000
## 197   14.80 0.00011    0.00000      0.00000   0.00000
## 198   14.85 0.00009    0.00000      0.00000   0.00000
## 199   14.90 0.00008    0.00000      0.00000   0.00000
## 200   14.95 0.00007    0.00000      0.00000   0.00000
## 201   15.00 0.00006    0.00000      0.00000   0.00000
```

```
# Devuelve una tabla con las columnas: pi_grid, prior, likelihood, unnormalized, posterior
```



Ejercicio 6.15 (MCMC con RStan: Gamma–Poisson)

Consideremos el modelo Gamma–Poisson para λ con

$$Y_i \mid \lambda \sim \text{Pois}(\lambda) \qquad \text{y} \qquad \lambda \sim \text{Gamma}(20,\, 5),$$

donde usamos la parametrización **Gamma(shape, rate)**. Observamos $n = 3$ datos independientes

$$(Y_1, Y_2, Y_3) = (0,\, 1,\, 0).$$

Objetivo

- Simular el modelo a posteriori de λ con **RStan** usando **4 cadenas** y **10 000 iteraciones por cadena**.
- Producir *trace plots* y densidades para las cuatro cadenas.
- A partir de las densidades, indicar el valor de λ más plausible a posteriori.
- Especificar el modelo a posteriori de λ por conjugación y comparar con la aproximación MCMC.

```
# Técnica de aproximación (MCMC) de la posteriori
# Produce cadenas dependientes de lambda a partir del modelo Gamma-Poisson.

# Ejemplo Gamma-Poisson
```

```
#
# PASO 1: Definición de la estructura del modelo bayesiano
gp_model <- "
  data {
    int<lower=0> N;            // número de observaciones
    int<lower=0> y[N];         // datos Poisson
    real<lower=0> alpha;       // shape de la Gamma prior
    real<lower=0> beta;        // rate  de la Gamma prior
  }
  parameters {
    real<lower=0> lambda;      // parámetro Poisson
  }
  model {
    lambda ~ gamma(alpha, beta);  // prior Gamma(shape, rate)
    y ~ poisson(lambda);          // verosimilitud
  }
"
```

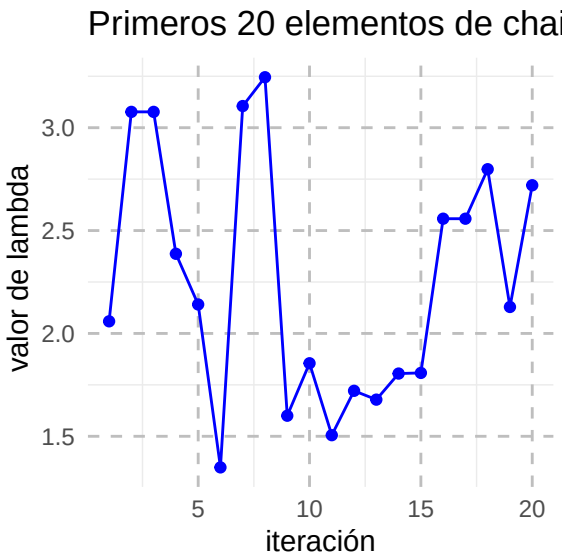
```
# Datos y prior
y <- c(0L, 1L, 0L)
N <- length(y)
alpha <- 20
beta  <- 5
```

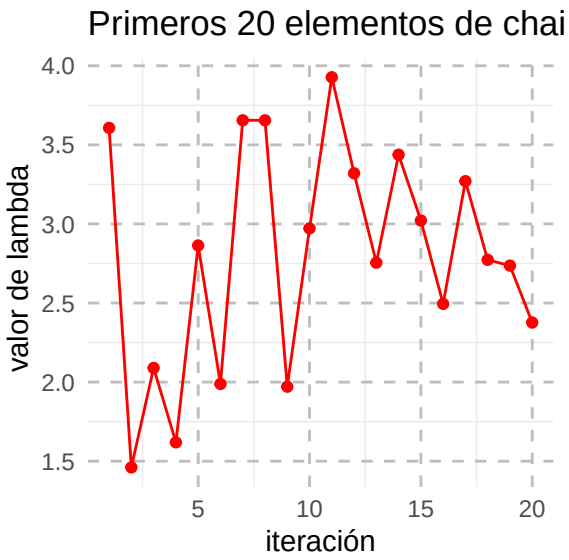
```
# PASO 2: Simulación de la distribución a posteriori con rstan::stan()
gp_sim <- stan(
  model_code = gp_model,
  data  = list(N = N, y = y, alpha = alpha, beta = beta),
  chains = 4, iter = 10000, warmup = 1000, seed = 84735
)
```

```
# Visualización de los primeros cuatro valores de las realizaciones de cada cadena
as.array(gp_sim, pars = "lambda") %>% head(4)
```

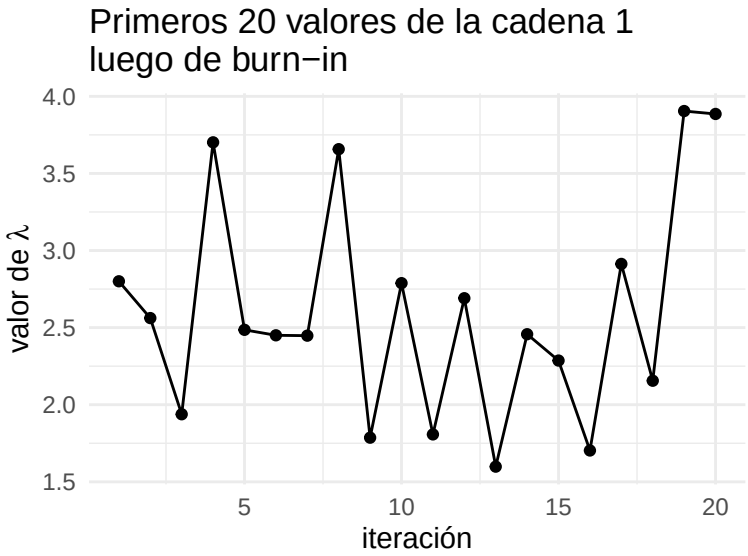
```
## , , parameters = lambda
##
##           chains
## iterations  chain:1  chain:2  chain:3  chain:4
##       [1,] 2.058805 2.953002 3.607447 3.039642
##       [2,] 3.076931 2.764217 1.460747 2.048070
##       [3,] 3.076931 2.764217 2.089549 1.884896
##       [4,] 2.387108 3.041081 1.618826 1.943340
```

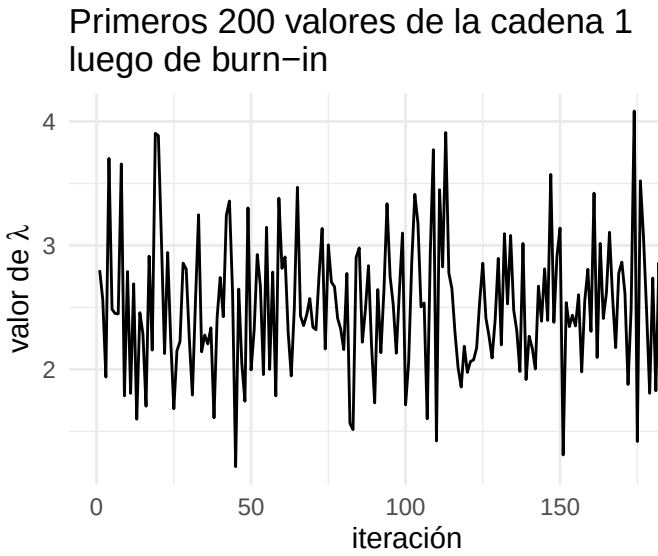
```
# Extraer los primeros 20 elementos de chain:1 y chain:3
samples_array <- as.array(gp_sim, pars = "lambda")
chain_1 <- samples_array[1:20, 1, 1]
chain_3 <- samples_array[1:20, 3, 1]
```



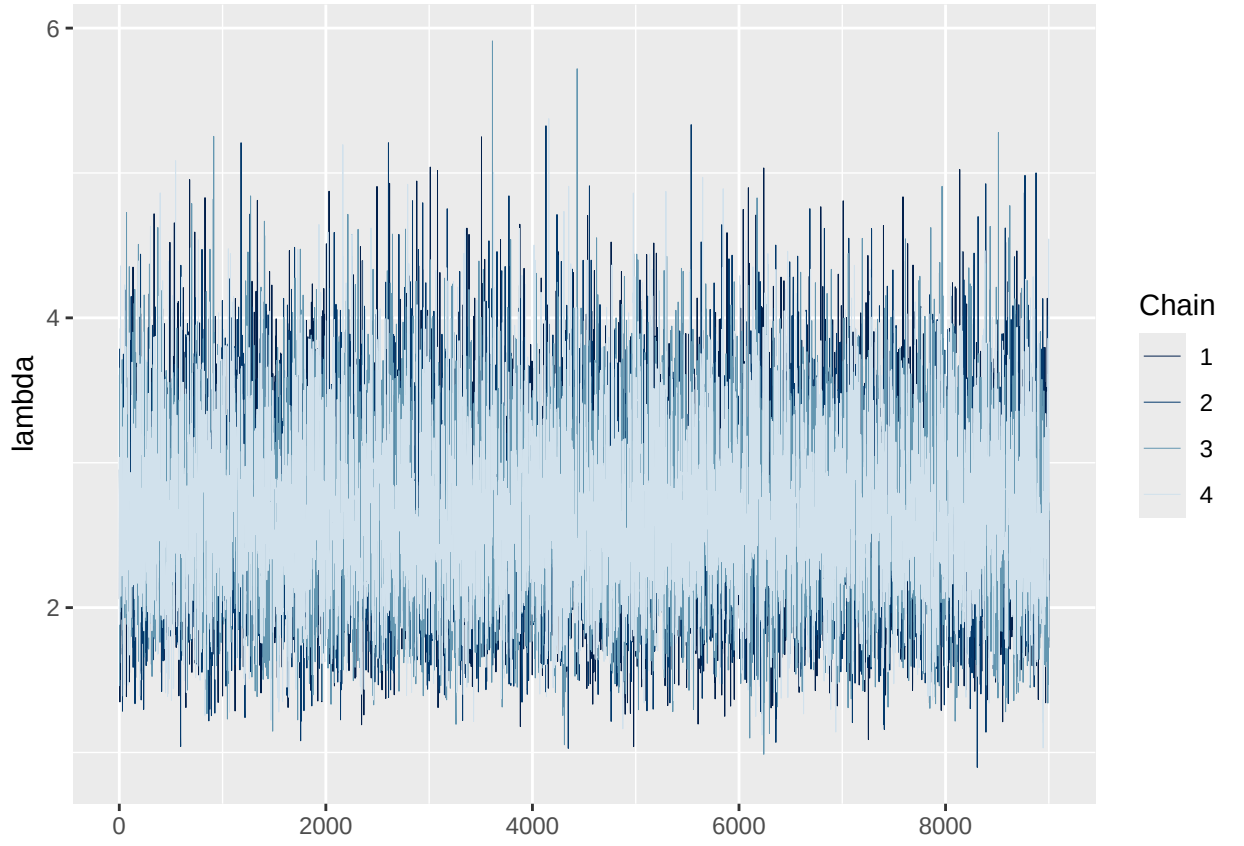



```
##  num [1:36000(1d)] 2.8 2.56 1.94 3.7 2.49 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ iterations: NULL
```



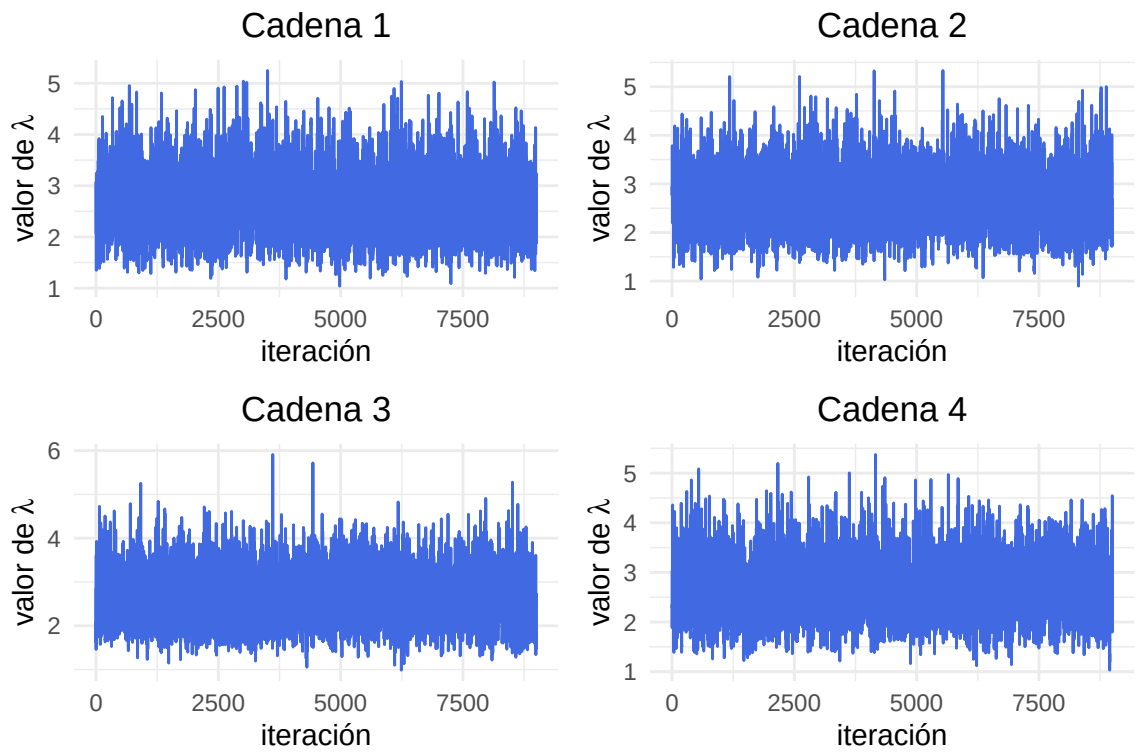



```
# Trazas de las cuatro cadenas de Markov para lambda
mcmc_trace(gp_sim, pars = "lambda", size = 0.1)
```

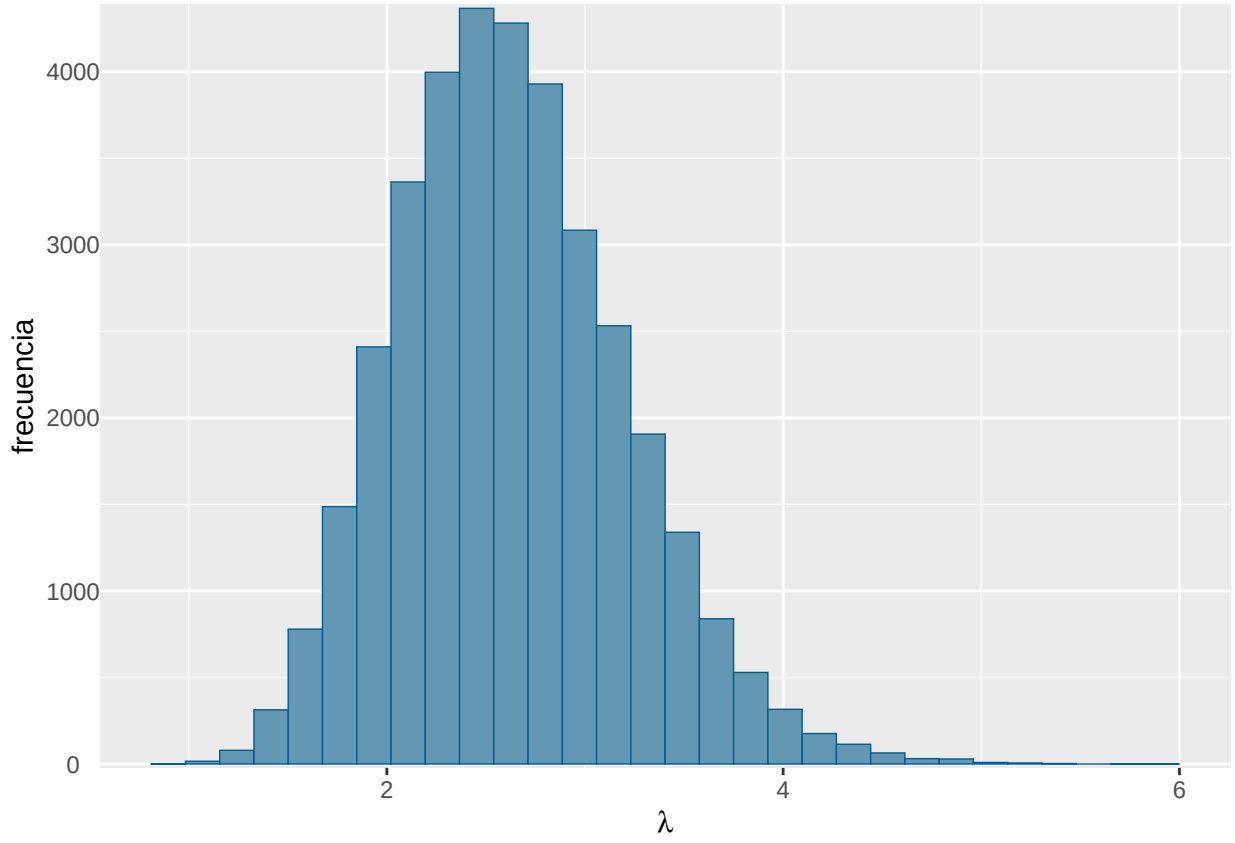


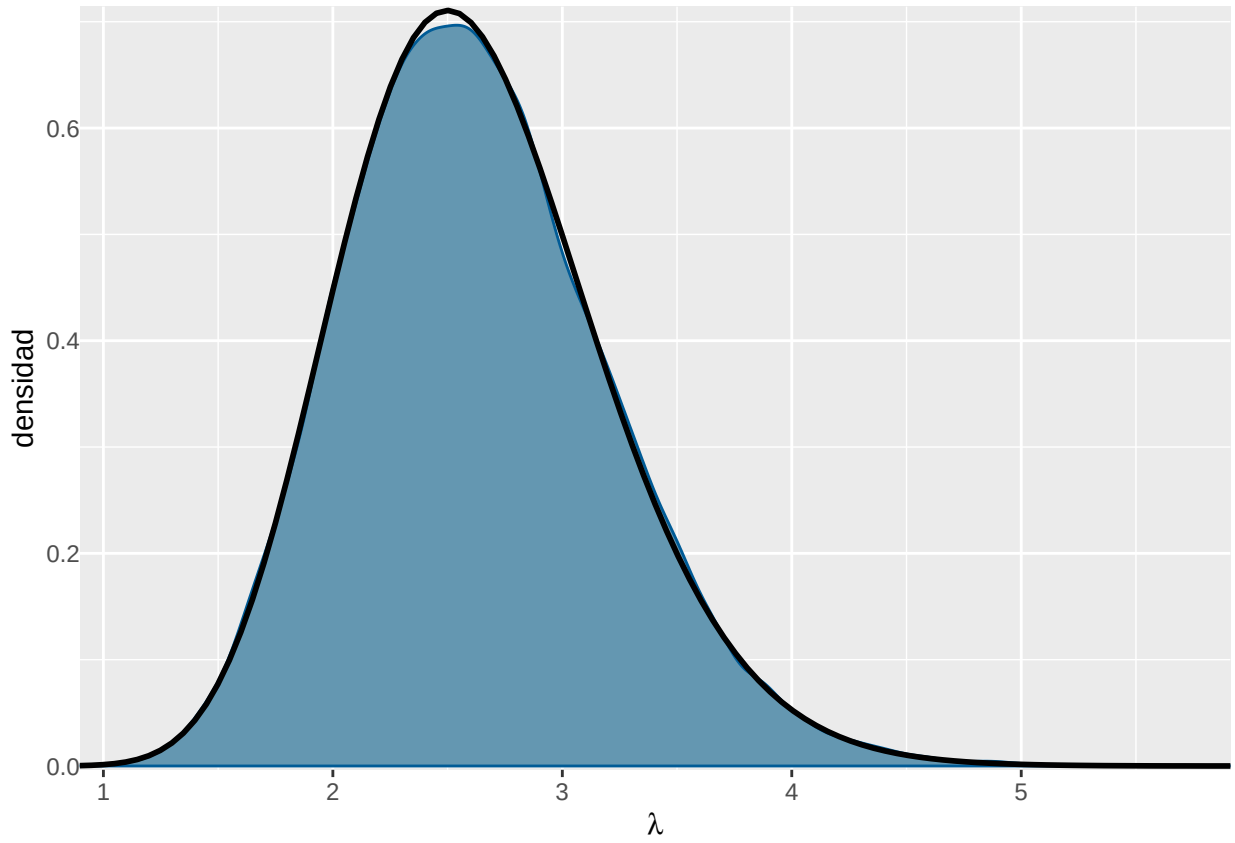

```r
# Extraer todas las cadenas completas (array)
samples_array <- as.array(gp_sim, pars = "lambda")
chain_1 <- samples_array[, 1, 1]
chain_2 <- samples_array[, 2, 1]
chain_3 <- samples_array[, 3, 1]
chain_4 <- samples_array[, 4, 1]
```

```r
# Función para graficar una cadena
grafico_cadena <- function(valores, titulo) {
  ggplot(data.frame(iteracion = 1:length(valores), lambda = valores),
         aes(x = iteracion, y = lambda)) +
    geom_line(color = "royalblue") +
    labs(title = titulo, x = "iteración", y = expression("valor de " * lambda)) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))
}
```

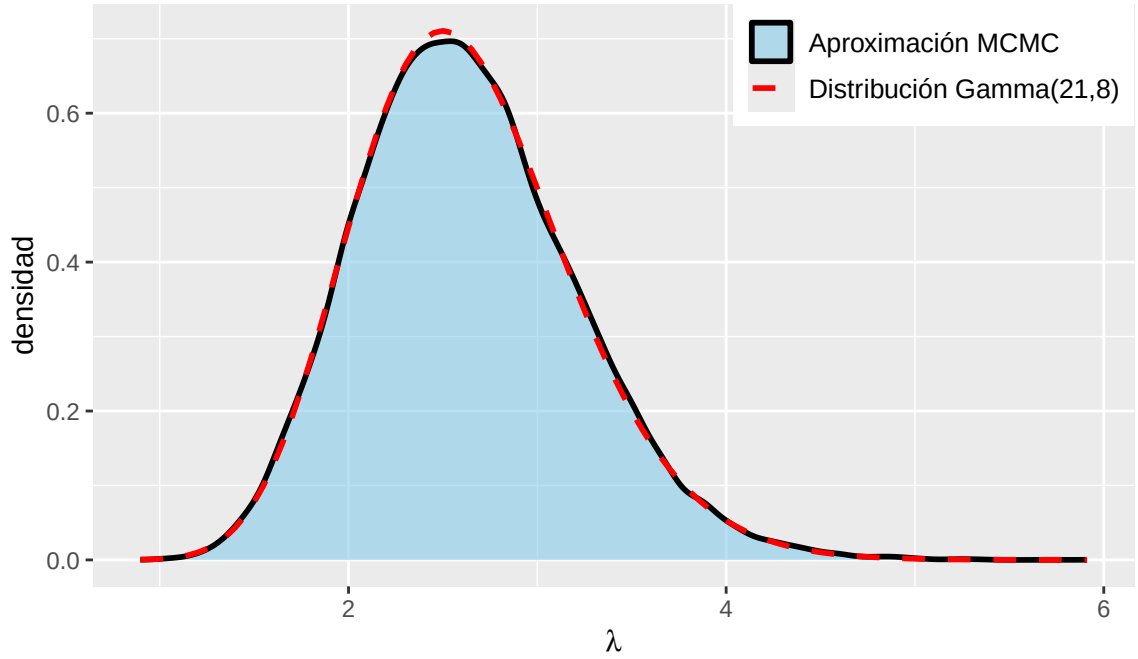



```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

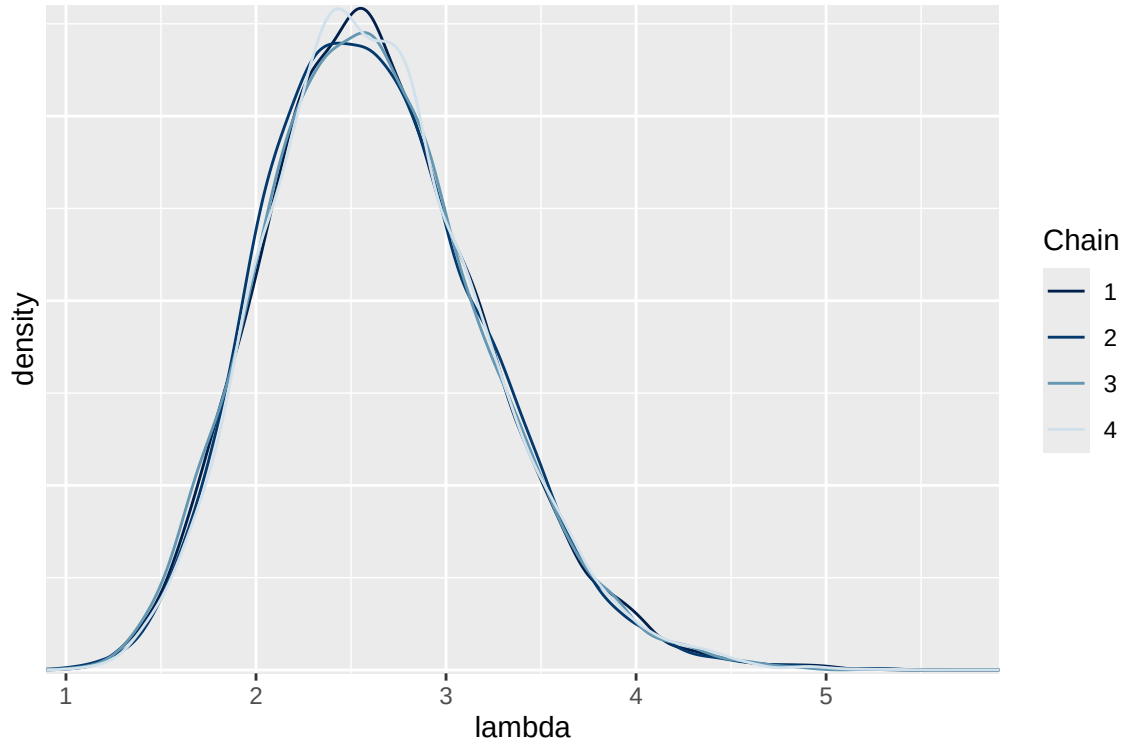




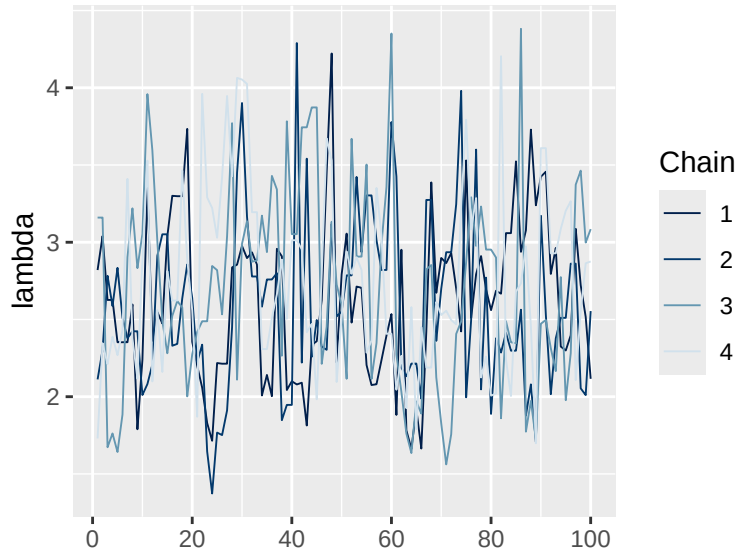
Comparación entre la aproximación MCMC de la posterior de .
y la distribución Gamma(21,8)



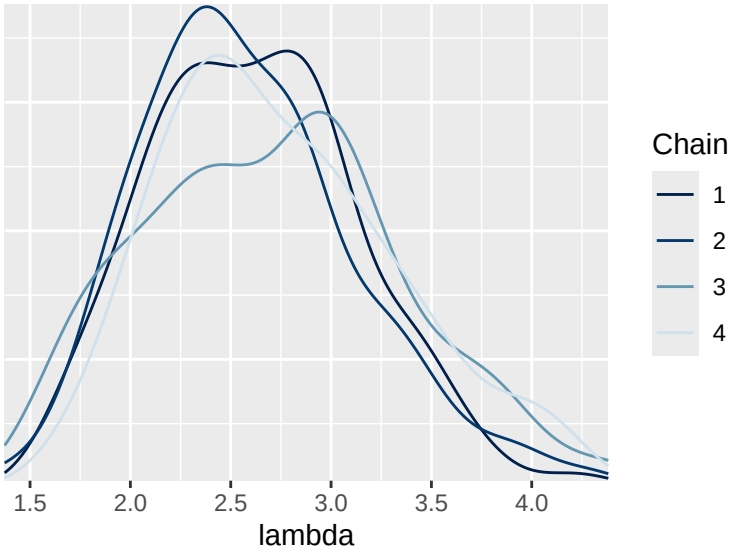
```r
# Densidades por cadena superpuestas
mcmc_dens_overlay(gp_sim, pars = "lambda") + ylab("density")
```




```r
# Simulación corta para ilustrar convergencia pobre
gp_sim_short <- stan(model_code = gp_model,
                     data = list(N = N, y = y, alpha = alpha, beta = beta),
                     chains = 4, iter = 100 * 2, seed = 84735)

mcmc_trace(gp_sim_short, pars = "lambda")
```



```r
mcmc_dens_overlay(gp_sim_short, pars = "lambda")
```



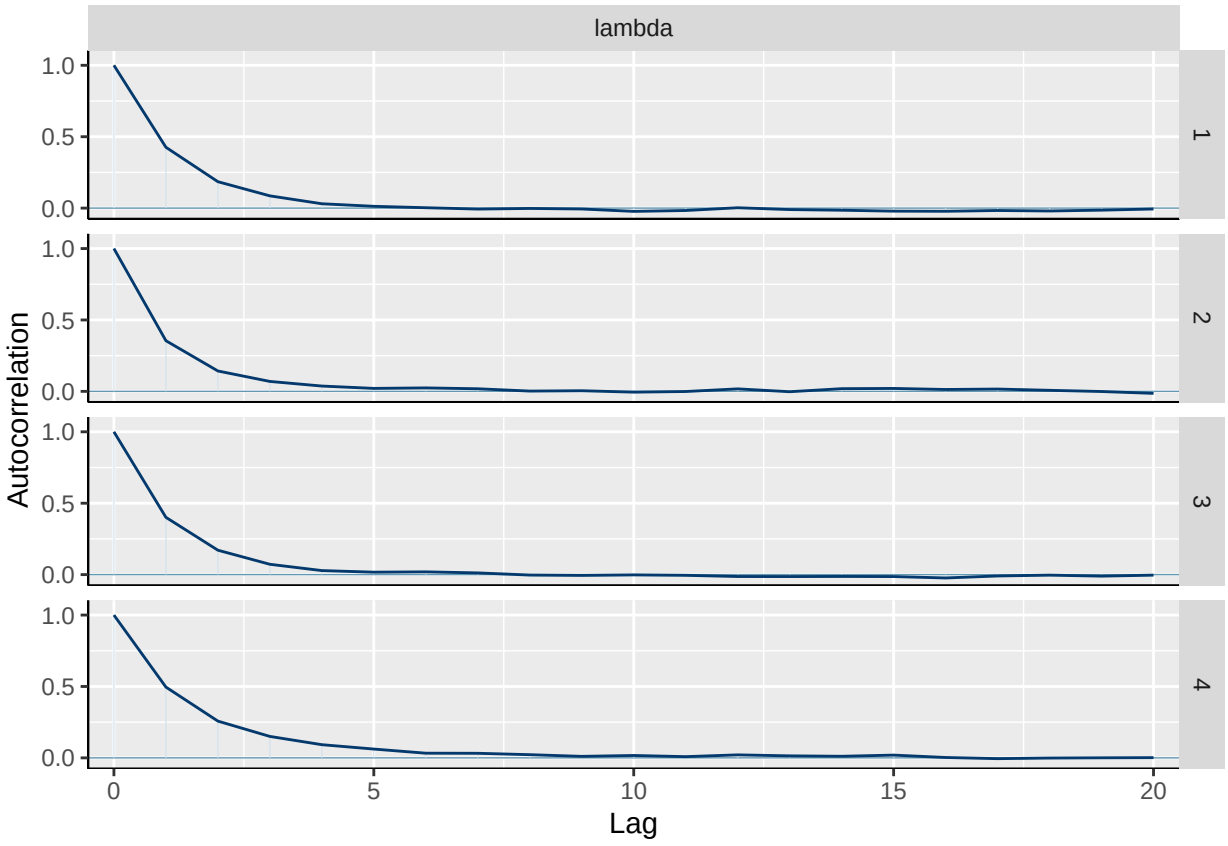

```r
# Diagnósticos: tamaño efectivo, R-hat y autocorrelación
neff_ratio(gp_sim, pars = c("lambda"))
```

```
## [1] 0.3802014
```

```r
rhat(gp_sim, pars = "lambda")
```

```
## [1] 1.000029
```

```r
mcmc_acf(gp_sim, pars = "lambda")
```



Condiciones iniciales para el capitulo 7

7.9.2 Práctica: Simulación Normal-Normal

En el siguiente conjunto de ejercicios, volvemos al modelo bayesiano del apartado (7.1):

$$Y \mid \mu \sim N(\mu, 0.75^2), \quad \mu \sim N(0, 1^2)$$

Supongamos que observamos Y = 6.25 y deseamos construir una simulación de Metropolis–Hastings del modelo posterior correspondiente para λ

Ejercicio 7.9 (Una iteración con un modelo de propuesta Uniforme)

La función `one_mh_iteration()` del texto utiliza un modelo de propuesta Uniforme:

$$\mu' \mid \mu \sim \text{Unif}(\mu - w,\, \mu + w)$$

con ancho w=1.

Partiendo de un valor actual de $\mu = 3$ y utilizando `set.seed(1)`, ejecuta el siguiente código y comenta sobre los valores retornados de `proposal`, `alpha`, y `next_stop`:

```r
one_mh_iteration(w = 0.01, current = 3)
one_mh_iteration(w = 0.5, current = 3)
one_mh_iteration(w = 1, current = 3)
one_mh_iteration(w = 3, current = 3)
```

Primero seteamos la semilla en 1 y aplicamos la función `one_mh_iteration()` al valor inicial $\mu = 3$ con un ancho de $w = 0.01$.

proposal	alpha	next_stop
3.0026	1	3.0026

Con nuestros primeros valores, el algoritmo partió de $\mu = 3$ y, dado que el ancho era muy pequeño ($w = 0.01$), la propuesta resultó ser apenas distinta: $\mu' = 3.0026$. Al comparar la plausibilidad del valor actual y de la propuesta bajo la posterior, ambas resultaron prácticamente iguales, de modo que la razón de aceptación alcanzó el valor máximo posible, $\alpha = 1$. Eso implica que la propuesta se acepta sin ninguna duda, y por lo tanto la cadena se mueve al nuevo valor, quedando en $\mu = 3.0026$. Esto tiene lógica porque, al ser la propuesta muy cercana al valor actual, tanto la prior como la verosimilitud cambian mínimamente, lo que hace que la posterior apenas varíe y, por ende, la aceptación sea prácticamente segura.

Ahora aplicamos la función `one_mh_iteration()` al mismo valor inicial $\mu = 3$, pero aumentando el ancho a $w = 0.5$.

proposal	alpha	next_stop
2.98	0.95	2.98

A diferencia de la iteración anterior con $w = 0.01$, donde la propuesta estaba prácticamente pegada al valor actual, esta vez la propuesta $\mu' = 2.98$ se aleja un poco más de $\mu = 3$. Como resultado, la plausibilidad bajo la posterior cambia un poco más, y la razón de aceptación $\alpha = 0.95$ ya no es 1, aunque sigue siendo alta. Esto significa que la propuesta se acepta con una probabilidad cercana a 1, lo que refleja que la cadena puede moverse pasos más grandes cuando el ancho es mayor. La lógica detrás de esto es permitirnos explorar un rango más amplio alrededor del valor actual, la cadena puede realizar saltos más amplios, manteniendo el equilibrio entre explorar nuevas regiones y quedarse en zonas de alta probabilidad, lo que nos asegura una buena exploración del espacio posterior.

A continuación, usamos `one_mh_iteration()` con el mismo valor inicial $\mu = 3$, pero aumentando aún más el ancho a $w = 1$.

proposal	alpha	next_stop
2.87	0.67	3

En comparación con las iteraciones anteriores, la propuesta $\mu' = 2.87$ se aleja bastante del valor actual. Esto provoca un cambio más marcado en la plausibilidad bajo la posterior, y la razón de aceptación cae a $\alpha = 0.67$. Como resultado, la propuesta no se acepta en esta ocasión, y la cadena permanece en $\mu = 3$. La lógica de este comportamiento es consistente con el algoritmo, al ampliar el rango de propuestas, se permiten saltos más grandes que exploran más el espacio, pero también aumenta la probabilidad de proponer valores menos compatibles con los datos y la prior. Así, el algoritmo equilibra exploración y aceptación, evitando moverse automáticamente a cualquier propuesta que no sea suficientemente plausible.

Finalmente, aplicamos `one_mh_iteration()` con el valor inicial $\mu = 3$ y un ancho aún mayor, $w = 3$.

proposal	alpha	next_stop
2.63	0.29	3

En esta iteración, la propuesta $\mu' = 2.63$ se aleja significativamente del valor actual, mucho más que en las pruebas anteriores. Esto provoca un cambio considerable en la plausibilidad de la posterior, y la razón de aceptación disminuye notablemente a $\alpha = 0.29$. Como resultado, la propuesta no se acepta, y la cadena permanece en $\mu = 3$. La lógica detrás de este comportamiento es coherente con el diseño del algoritmo: al generar propuestas que se alejan demasiado del valor actual, la probabilidad de aceptar un valor menos compatible con los datos y la prior disminuye, lo que evita saltos grandes que podrían llevar a regiones de baja probabilidad.

En pocas palabras, con w pequeño la cadena avanza despacio y casi siempre acepta, mientras que con w más grande las propuestas se alejan más y se aceptan menos. Elegir w es un equilibrio entre pasos seguros y explorar más el espacio.

Ejercicio 7.10 (Un recorrido completo con un modelo de propuesta Uniforme)

Implementa la función de Metropolis-Hastings `mh_tour()` definida en la Sección 7.3 para construir recorridos de μ bajo cada uno de los siguientes escenarios. Construye gráficos de trazas e histogramas para cada recorrido:

1. 50 iteraciones, $w = 50$