Modelo Lineal Bayesiano

1)

Sea Y una Variable Aleatoria modelable con una distribucion Geométrica.

$$Y|p \sim Ge(p)$$

$$P(Y = y|\theta) = L_{Y=y}(\theta) = \theta(1-\theta)^{y-1}$$

El prior del parametro θ esta definido por una distribución Beta.

$$\theta \sim Beta(\alpha, \beta)$$

$$f(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha).\Gamma(\beta)}.\theta^{\alpha-1}.(1-\theta)^{\beta-1}$$

a.

Una variable aleatoria geométrica puede representar situaciones en las cuales uno este interesado en cuando sucede el primer éxito de un experimento. Siendo mas precisos, dada $Y \sim Ge(p)$, la variable aleatoria Y modela el numero de fracasos antes del primer exito de un experimento con probabilidad de exito p .

b.

Dado $Y = y$, derivo la distribución posterior.

$$f(\theta|Y = y) = \frac{1}{\int_R L_{Y=y}(\theta).f(\theta)d\theta}.L_{Y=y}(\theta).f(\theta)$$

$$f(\theta|Y = y) \propto L_{Y=y}(\theta).f(\theta)$$

$$L_{Y=y}(\theta).f(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha).\Gamma(\beta)}.\theta^{\alpha-1}.(1-\theta)^{\beta-1}.\theta(1-\theta)^{y-1}$$

$$L_{Y=y}(\theta).f(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha).\Gamma(\beta)}.\theta^{\alpha}.(1-\theta)^{\beta+y-2}$$

Lo cual da como resultado el mismo nucleo que una beta, dandola siguiente distribucion de posterior.

$$f(\theta|Y = y) \sim Beta(\alpha+1, \beta+y-1)$$

c.

Para que una distribución sea un priori conjuado de otra, debe pasar que la distribución posterior, $f(\theta|Y = y)$ es de la misma familia de distribuciones que la priori $f(\theta)$, lo cual es el caso con la distribución Beta y la distribución Geometrica.

2)

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Una posible pregunta que se puede responder con un modelo de regresion lineal es cual es la eficiencia de un motor en función de su tamaño.

$$Y_i \sim N(\mu_n, \sigma)$$

$$\mu_n = \beta_0 + \beta_1 x_n$$

El dataframe fue conseguido en kaggle.

En este dataframe, "city_mpg", "highway_mpg", y "combined_mpg" representan las millas las cuales el auto es capaz de recorrer por galon de combustible gastado, esta medida es realizada en la ciudad y en autopista respectivamente, mientras que como el nombre sugiere nuestra ultima variable es una metrica compuesta. Por otro lado "displacement" mide el volumen de liquido que puede contener el motor, por lo cual es comunmente usado para medir el tamaño de este.

Intentaremos de predecir la economia del combustible de nuestro auto a traves del tamaño del motor, para esto sera necesario ignorar el tamaño del auto por lo cual se seleccionara una unica categoria

```r
car_data <-read.csv('car_data.csv')
colnames(car_data)
```

```
##  [1] "city_mpg"        "class"           "combination_mpg" "cylinders"
##  [5] "displacement"    "drive"           "fuel_type"       "highway_mpg"
##  [9] "make"            "model"           "transmission"    "year"
```

Limpiemos nuestros datos para poder usarlos.

```r
car_data<-select(filter(car_data, class=="midsize car"), combination_mpg, displacement)
```

a.

Los priors que propongo son los siguientes

$$\beta_0 \sim N(43, 5)$$

$$\beta_1 \sim N(-5.5, 2)$$

$$\sigma \sim Exp(\frac{1}{2})$$

b.

Dada una muestra aleatoria $M = \{V_1, V_2, ..., V_n\}$ donde $V_i = (y_i, x_i)$ la Likelyhood de nuestros datos esta dada por la siguiente formula.

$$L_M(\beta_0, \beta_1, \sigma) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-1}{2\sigma^2}(y_i - \beta_0 - \beta_1.x_i)^2)$$

c.

```
logLikelihood <- function(beta_0, beta_1, sigma, x, y) {
  mu <- beta_0 + beta_1 * x
  sum(dnorm(y, mean = mu, sd = sigma, log = TRUE))
}






logPlaus <- function(beta_0, beta_1, sigma, x, y) {
  return(log(pBeta0(beta_0))
        + log(pBeta1(beta_1))
        + log(pSigma(sigma))
        + logLikelihood(beta_0, beta_1, sigma, x, y))
}






metropolis<- function(sample_size, y, x, starting_point) {

  beta_0 <- starting_point[1]
  beta_1 <- starting_point[2]
  sigma <- starting_point[3]


  chain_beta_0 <- rep(0, sample_size)
  chain_beta_1 <- rep(0, sample_size)
  chain_sigma <- rep(0, sample_size)


  for (i in 1:sample_size) {


    beta_0_proposal <- rnorm(1, beta_0, 0.2)
    beta_1_proposal <- rnorm(1, beta_1, 0.2)
    sigma_proposal <- abs(rnorm(1, sigma, 0.2))

    log_current <- logPlaus(beta_0, beta_1, sigma, x, y)
    log_proposal <- logPlaus(beta_0_proposal, beta_1_proposal, sigma_proposal, x, y)

    p <- min(1, exp(log_proposal - log_current))

    if (sample(c(TRUE, FALSE), size = 1, prob = c(p, 1-p))) {
      beta_0 <- beta_0_proposal
      beta_1 <- beta_1_proposal
      sigma <- sigma_proposal
    }

    chain_beta_0[i] <- beta_0
    chain_beta_1[i] <- beta_1
    chain_sigma[i] <- sigma
  }

  chains <- data.frame (
    beta_0 = chain_beta_0,
    beta_1 = chain_beta_1,
    sigma = chain_sigma
  )
  return(chains)
}
```

Defino mis priors y mi punto de inicio.

```
pBeta0 <- function(beta_0){
  return(dnorm(beta_0, 43, 5))
  }
pBeta1 <- function(beta_1){
  return(dnorm(beta_1, -5.5, 2))
  }
pSigma <- function(sigma){
  return(dexp(sigma, 0.5))
}

starting_point<- c(0,0,1)
```

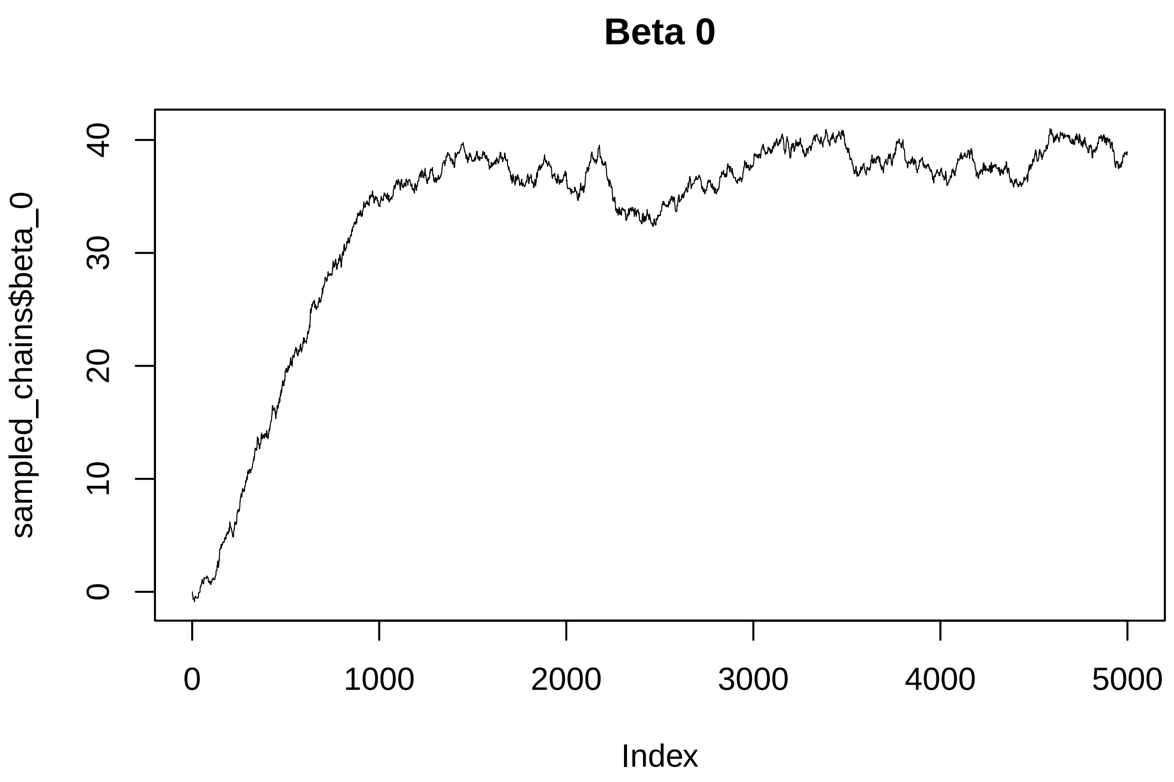
Utilizo el algoritmo Metropolis con 0 Burn, se estan tomando 10000 samples en vez de 5000 porque las cadenas de Beta1 son ligeramente mas volatiles:

```
y<-car_data$combination_mpg
x<-car_data$displacement

sampled_chains <- metropolis(5000, y, x, starting_point)
```

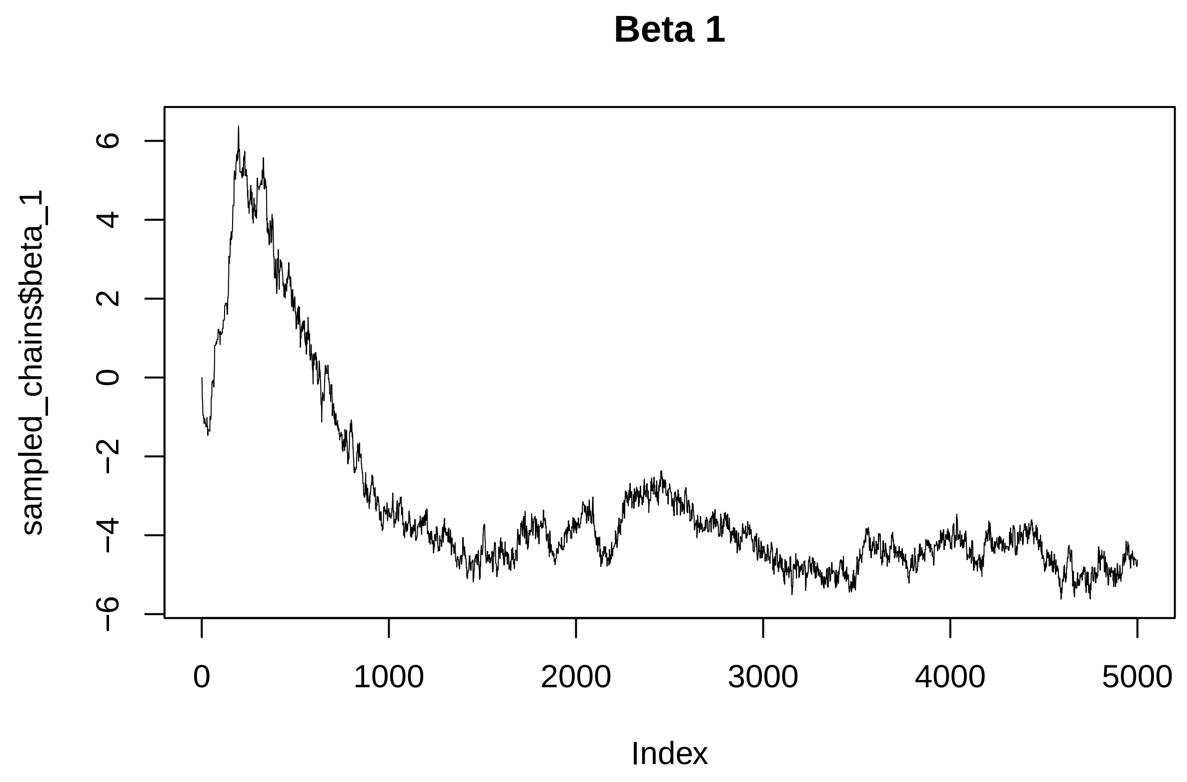
d.

Veo el grafico de las cadenas

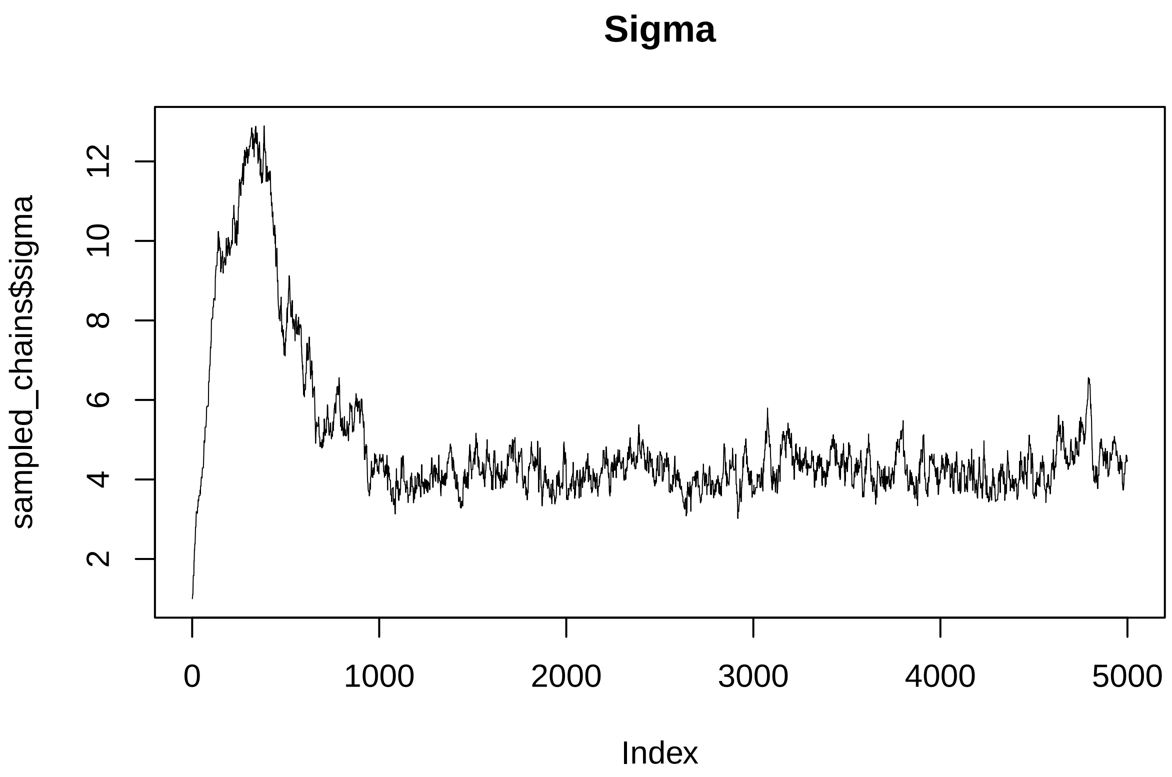
```
plot(sampled_chains$beta_0, type = "l", lwd=0.5, main="Beta 0")
```



```
plot(sampled_chains$beta_1, type = "l", lwd=0.5, main="Beta 1")
```



```r
plot(sampled_chains$sigma, type = "l", lwd=0.5, main="Sigma")
```

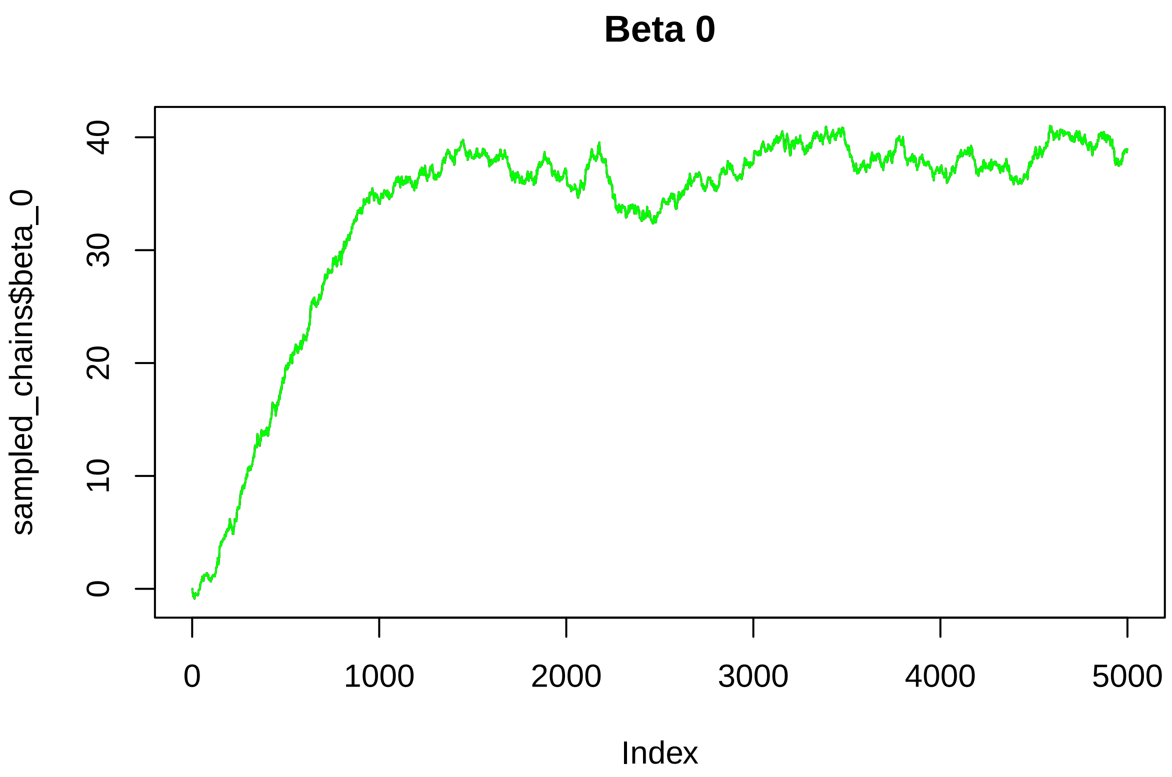


La cadena parece estabilizarse despues de 1.5k iteraciones. Los valores entre los que se mueve mas frequentemente cada parametro son entre 30 y 40 para beta 0, entre -3 y -5 para beta 1 y entre 3 y 5 para sigma.

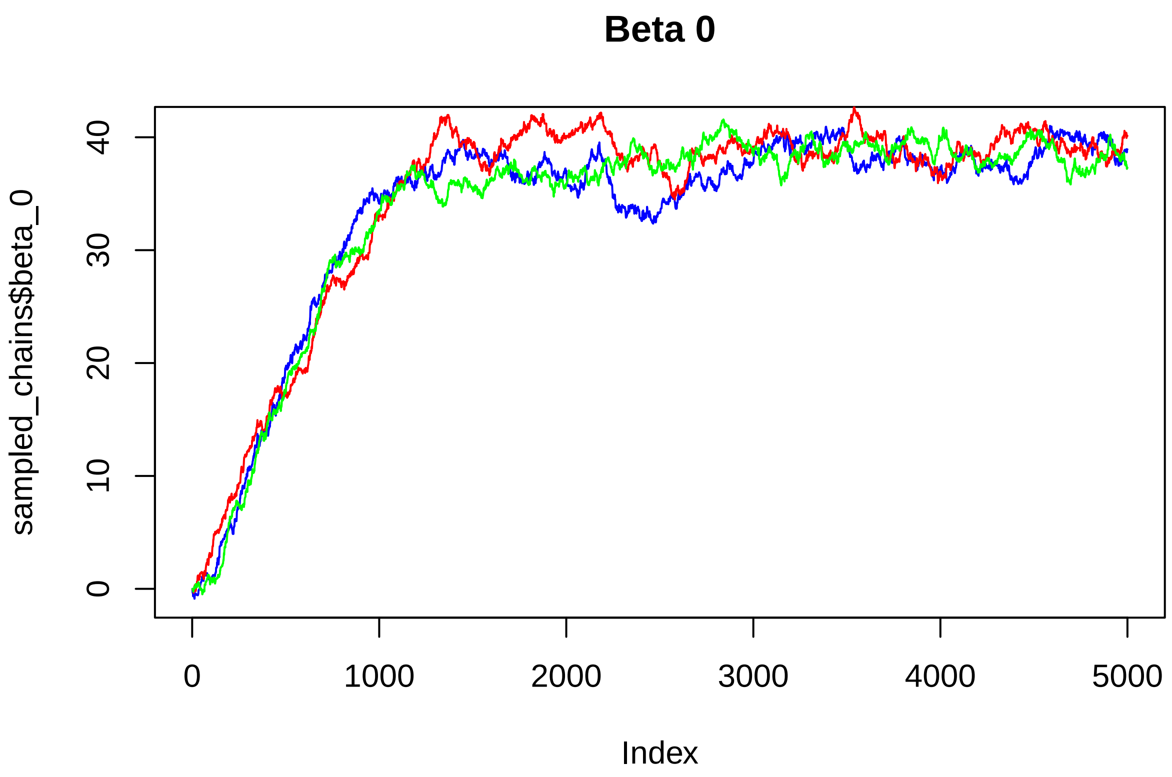
e.

Repetimos el proceso para 3 cadenas paralelas con distintos puntos de inicio.

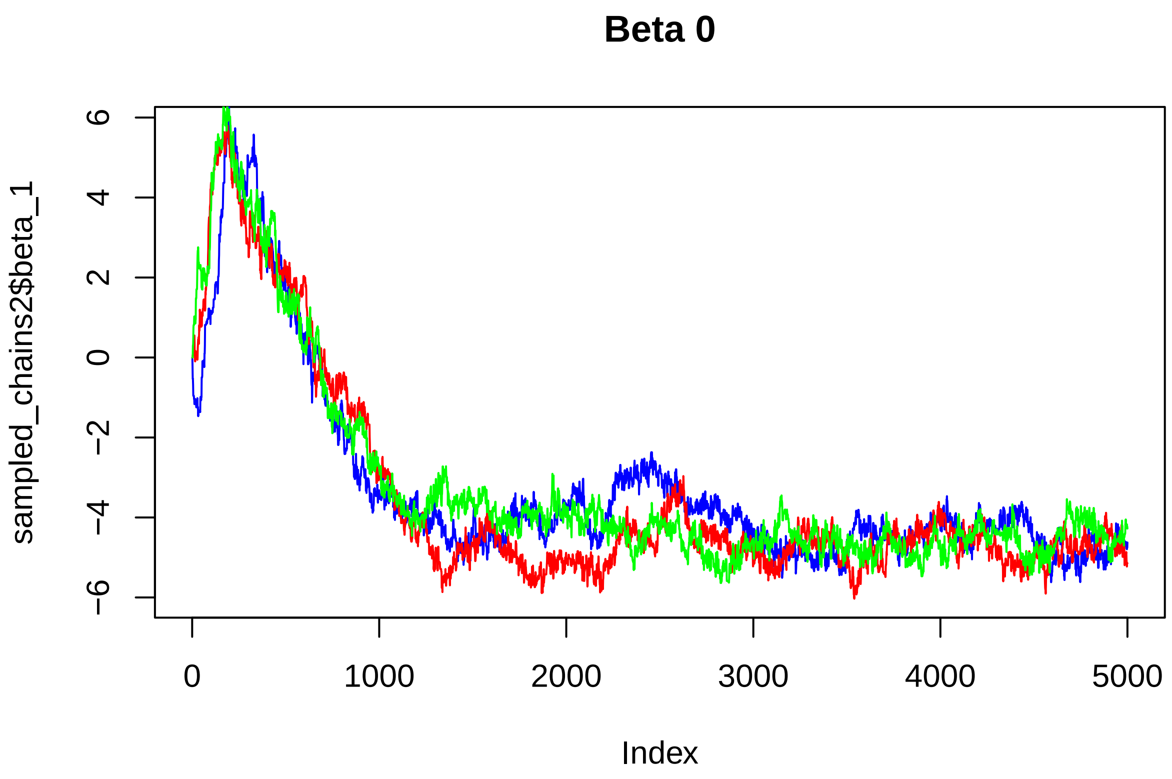
```r
plot(sampled_chains$beta_0, type = "n", main = "Beta 0")

lines(sampled_chains$beta_0, type = "l", col = "blue")
lines(sampled_chains$beta_0, type = "l", col = "red")
lines(sampled_chains$beta_0, type = "l", col = "green")
```



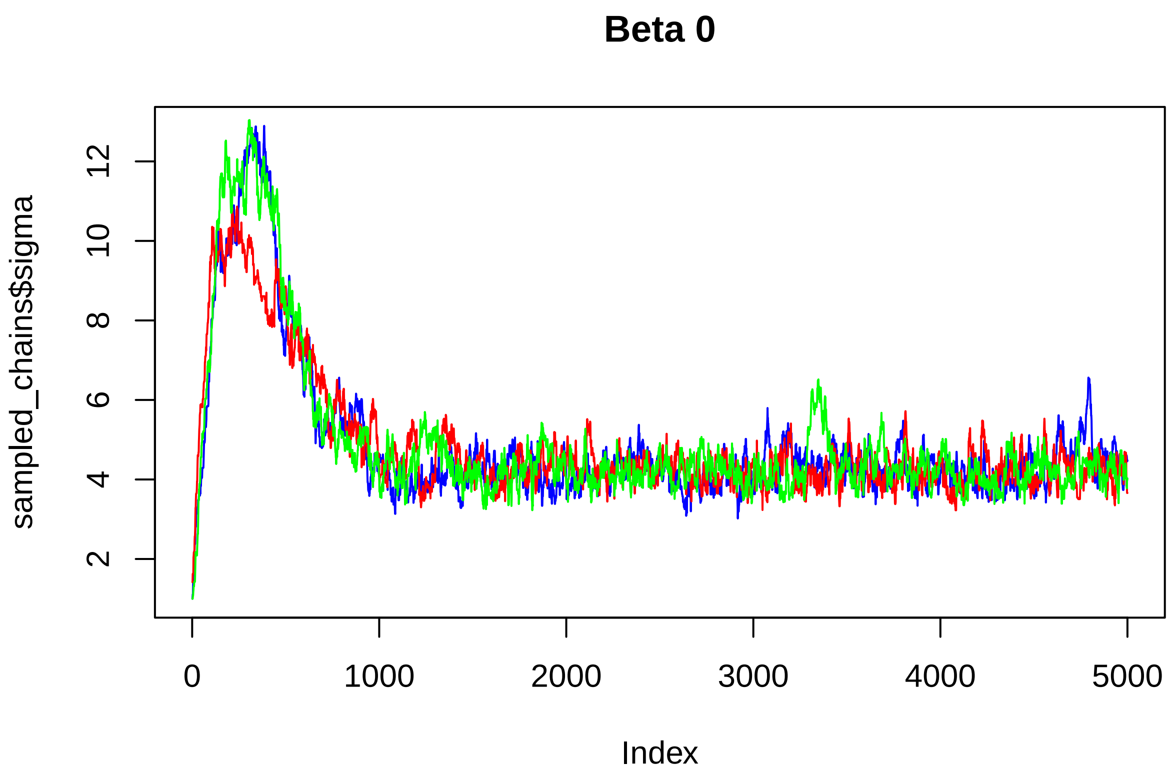
```r
starting_point2 <- c(100,-15,10)
sampled_chains2<- metropolis(5000, y, x, starting_point)

starting_point3 <- c(100,-15,10)
sampled_chains3<- metropolis(5000, y, x, starting_point)

# BETA 0
plot(sampled_chains$beta_0, type = "n", main = "Beta 0")

lines(sampled_chains$beta_0, type = "l", col = "blue")
lines(sampled_chains2$beta_0, type = "l", col = "red")
lines(sampled_chains3$beta_0, type = "l", col = "green")
```



```
# BETA 1
plot(sampled_chains2$beta_1, type = "n", main = "Beta 0")

lines(sampled_chains$beta_1, type = "l", col = "blue")
lines(sampled_chains2$beta_1, type = "l", col = "red")
lines(sampled_chains3$beta_1, type = "l", col = "green")
```



```r
# SIGMA
plot(sampled_chains$sigma, type = "n", main = "Beta 0")

lines(sampled_chains$sigma, type = "l", col = "blue")
lines(sampled_chains2$sigma, type = "l", col = "red")
lines(sampled_chains3$sigma, type = "l", col = "green")
```



Todas las cadenas llegan a un estado de equilibrio a partir de la iteracion 1500

f.

Elegimos 100 samples al azar.

```
burn<-2000
sample_size <- 100
positions_chosen<-sample(((burn+1):length(sampled_chains$beta_0)), sample_size)
sample<- data.frame (
    beta_0 = rep(0, sample_size),
    beta_1 = rep(0, sample_size),
    sigma = rep(0, sample_size)
  )
for(i in 1:sample_size){
  sample$beta_0[i]<-sampled_chains$beta_0[positions_chosen[i]]
  sample$beta_1[i]<-sampled_chains$beta_1[positions_chosen[i]]
  sample$sigma[i]<-sampled_chains$sigma[positions_chosen[i]]
}
```

Grafico las rectas superpuestas a los datos:

```
plot(x, y, xlab = "Displacement", ylab = "MPG", main = "Fit")
for(i in 1:100){
  lines(x, sample$beta_0[i] + sample$beta_1[i]*x, type = "l", col = "blue")
}
```

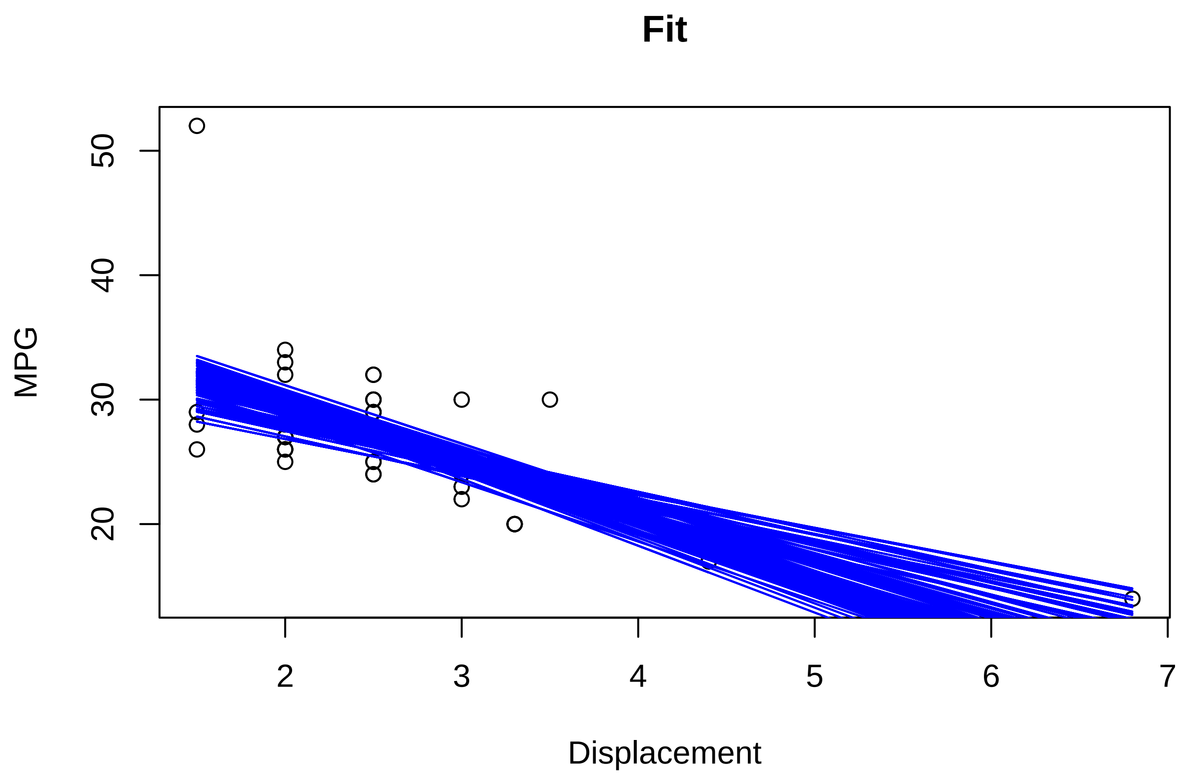
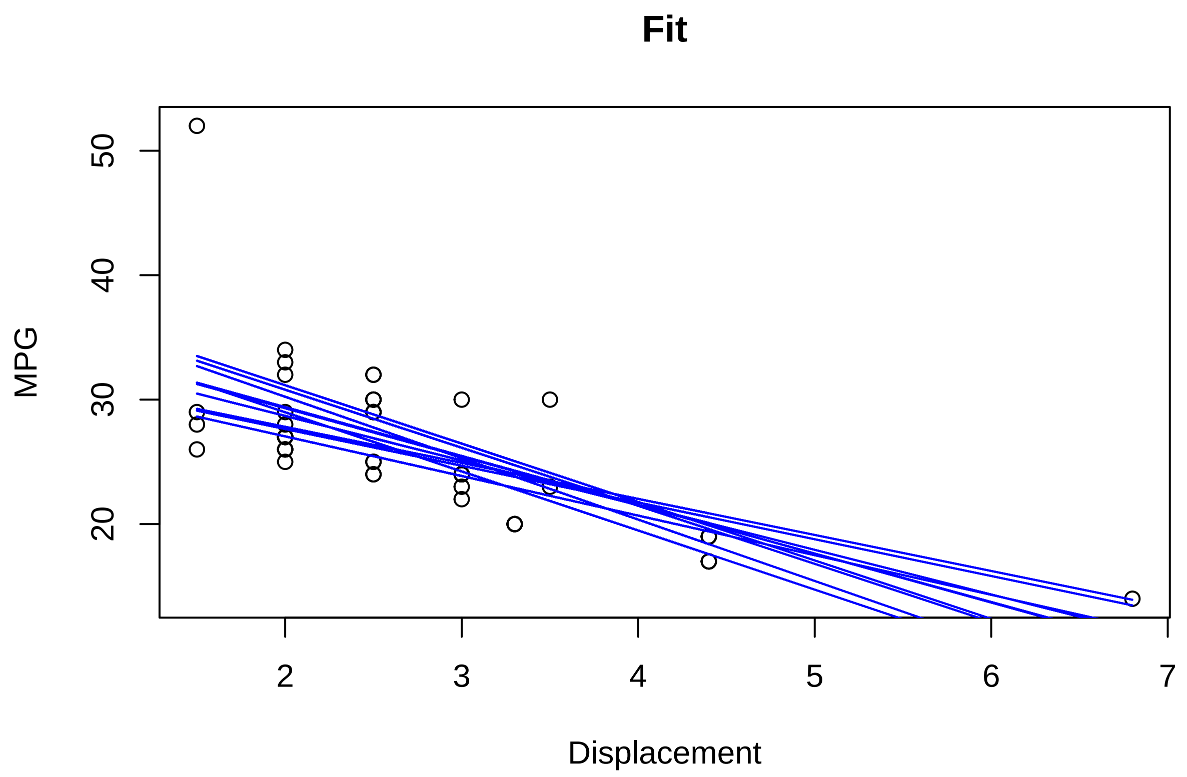


Grafico 10 para que se entienda mejor el grafico.

```
plot(x, y, xlab = "Displacement", ylab = "MPG", main = "Fit")
for(i in 1:10){
  lines(x, sample$beta_0[i] + sample$beta_1[i]*x, type = "l", col = "blue")
}
```

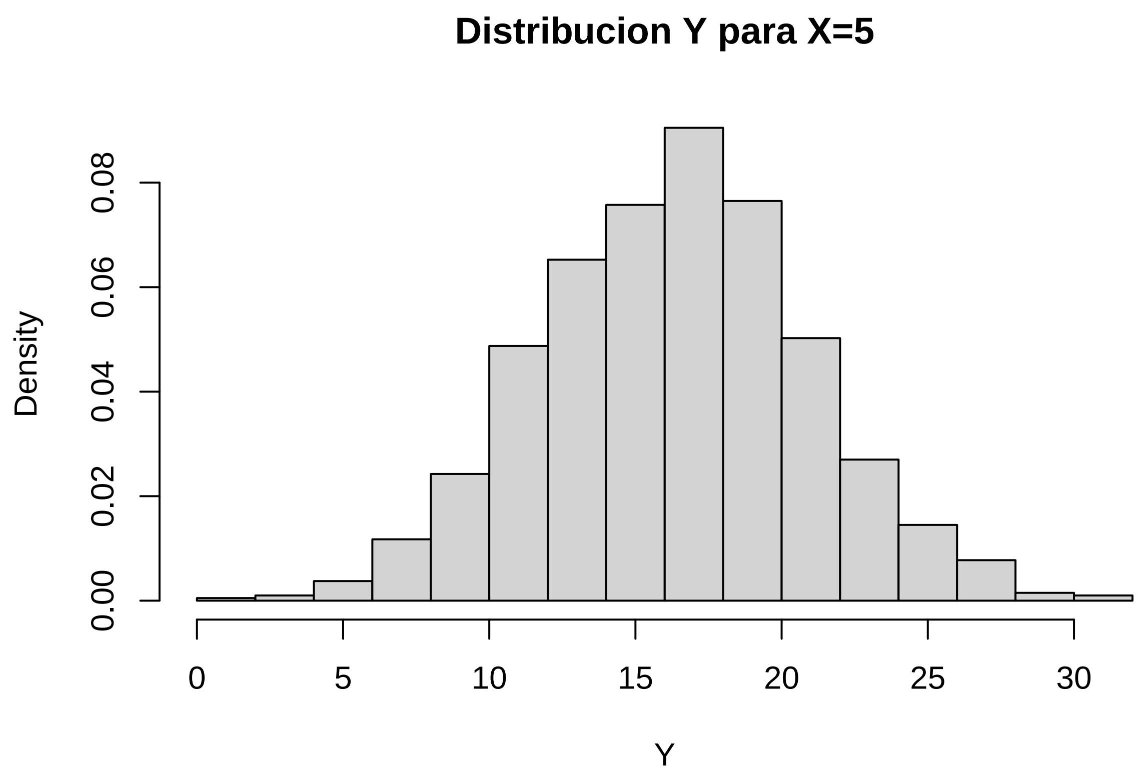


g.

```
PostPred<- function(burn, df, sample_size, x){
  positions_chosen<-sample((burn+1):length(df$beta_0), sample_size)
  res<- data.frame (
    beta_0 = rep(0, sample_size),
    beta_1 = rep(0, sample_size),
    sigma = rep(0, sample_size),
    y = rep(0, sample_size)
  )
  for(i in 1:sample_size){
    res$beta_0[i]<-df$beta_0[positions_chosen[i]]
    res$beta_1[i]<-df$beta_1[positions_chosen[i]]
    res$sigma[i]<-df$sigma[positions_chosen[i]]
    res$y[i]<-rnorm(1, mean = df$beta_0[positions_chosen[i]] + df$beta_1[positions_chosen[i]] * x, sd =
  }
  return(res)
}
```

Veamos la distribucion generada para un displacement de 5.

```r
param_sample <- PostPred(2000, sampled_chains, 2000, 5)
hist(param_sample$y, probability = TRUE, main = "Distribucion Y para X=5", xlab = "Y")
```



Con lo que podemos estimar que la esperanza de la autonomia para un auto con un motor de dicho tamaño es de 16 mpg.

3)

```r
library(brms)
```

```
## Loading required package: Rcpp

## Loading 'brms' package (version 2.22.2). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').

##
## Attaching package: 'brms'

## The following object is masked from 'package:stats':
##
##     ar
```

```r
priors <- prior(normal(43, 5), class = "b", coef = "Intercept") + prior(normal(-5.5, 2), class = "b", c

fit_gaussian <- brm(bf(combination_mpg ~ 1 + displacement, center = FALSE), data = car_data,
prior = priors)
```

```
## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.028 seconds (Warm-up)
## Chain 1:                0.028 seconds (Sampling)
## Chain 1:                0.056 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.027 seconds (Warm-up)
## Chain 2:                0.022 seconds (Sampling)
## Chain 2:                0.049 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.028 seconds (Warm-up)
## Chain 3:                0.026 seconds (Sampling)
## Chain 3:                0.054 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.025 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.046 seconds (Total)
## Chain 4:
```

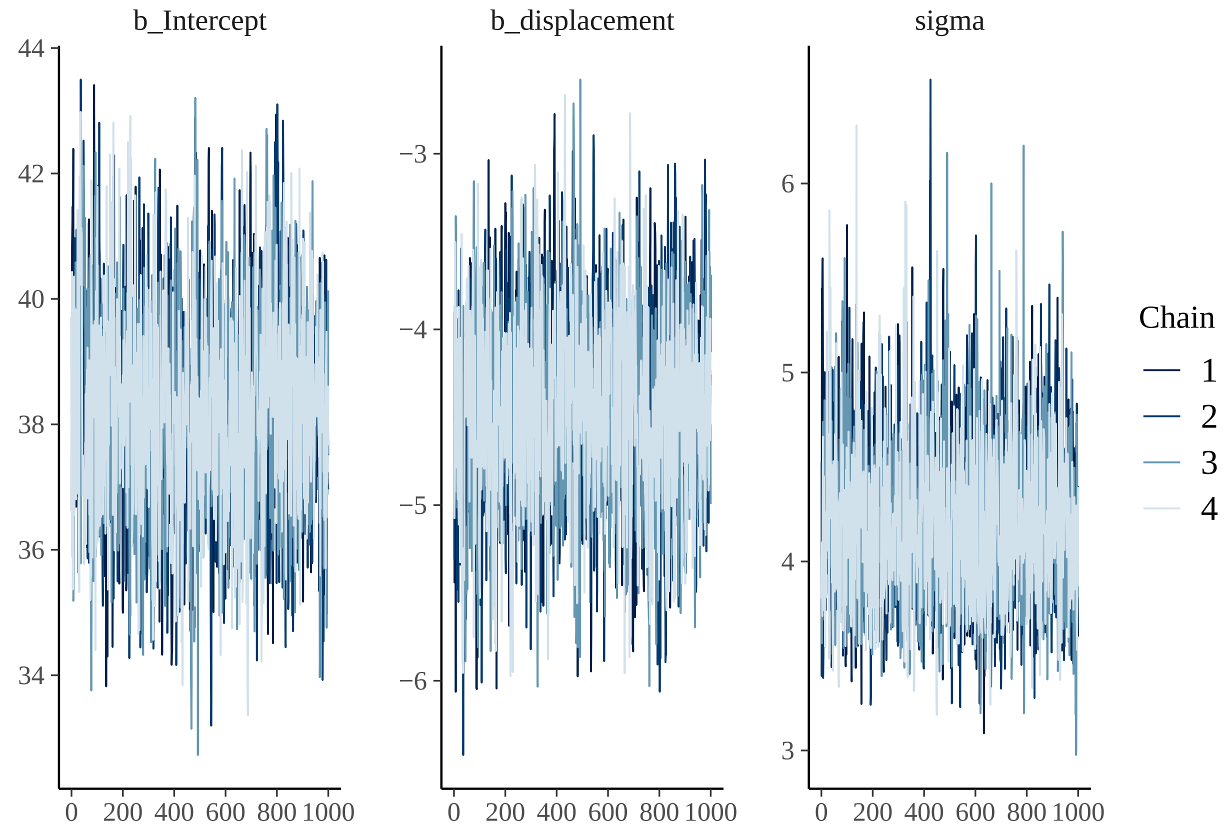
```r
summary(fit_gaussian)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: combination_mpg ~ 1 + displacement
##    Data: car_data (Number of observations: 53)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept       38.22      1.60    35.13    41.52 1.00     1437     1261
## displacement    -4.46      0.54    -5.55    -3.43 1.00     1407     1259
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.22      0.43     3.50     5.16 1.00     1946     1895
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

a.

```r
mcmc_plot(fit_gaussian, type = "trace")
```

```
## No divergences to plot.
```

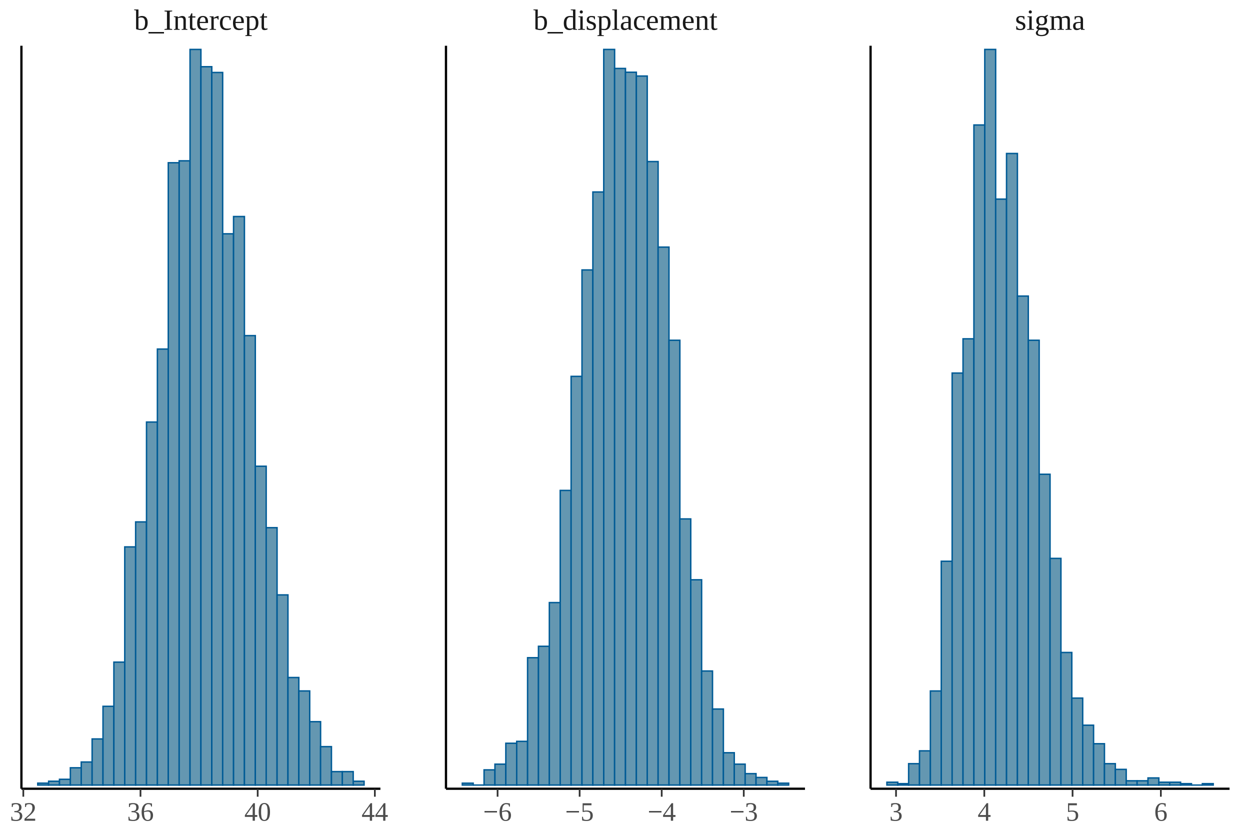


Parecieran haber convergido.

b.

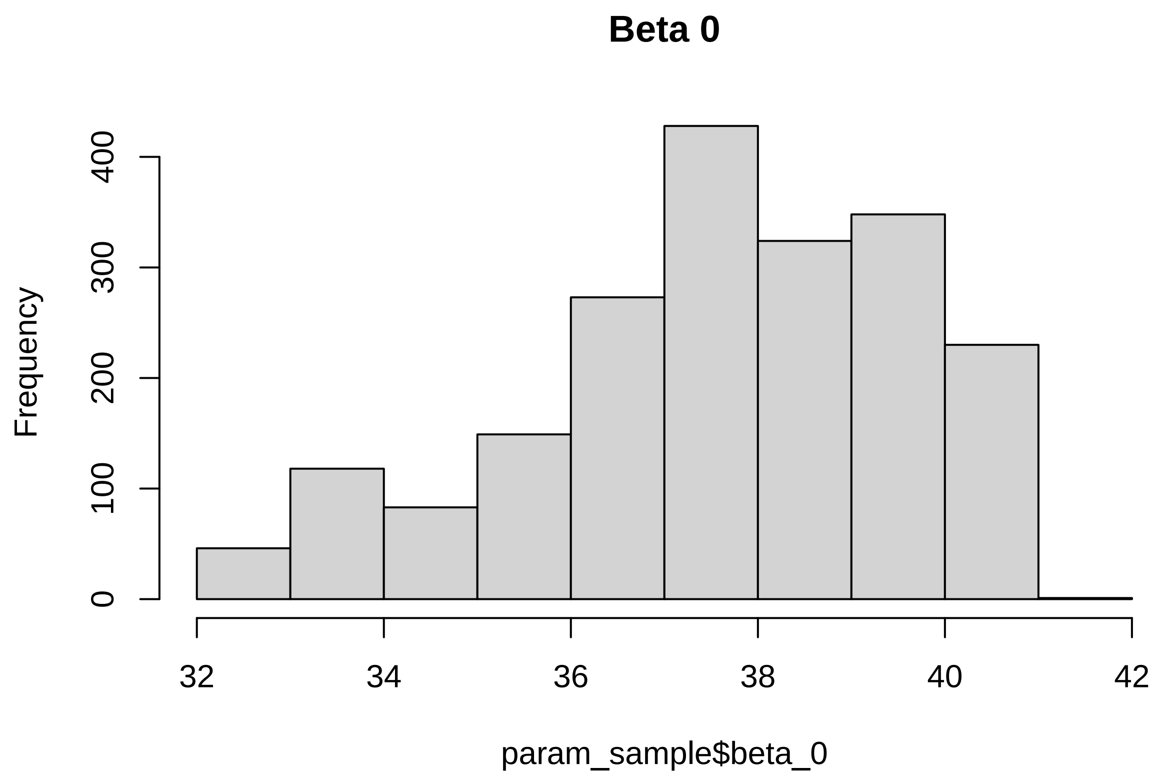
Samples de brms

```r
mcmc_plot(fit_gaussian, type = "hist", bins = 30)
```

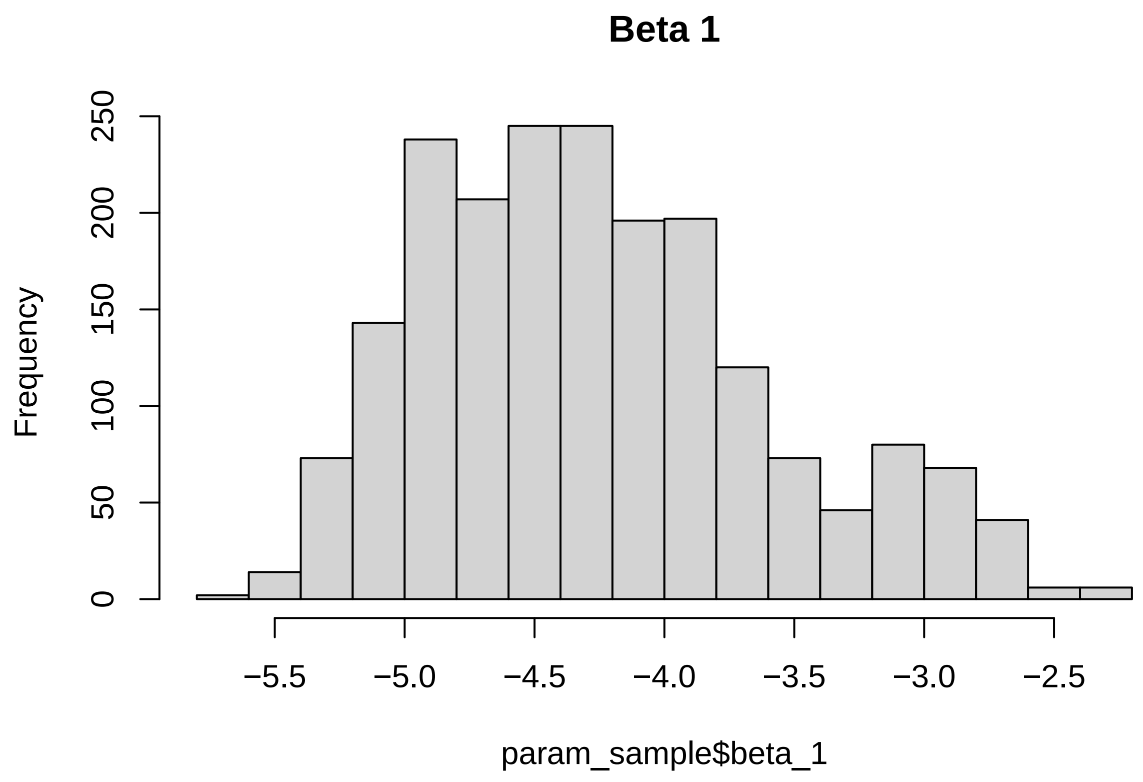


Samples propios

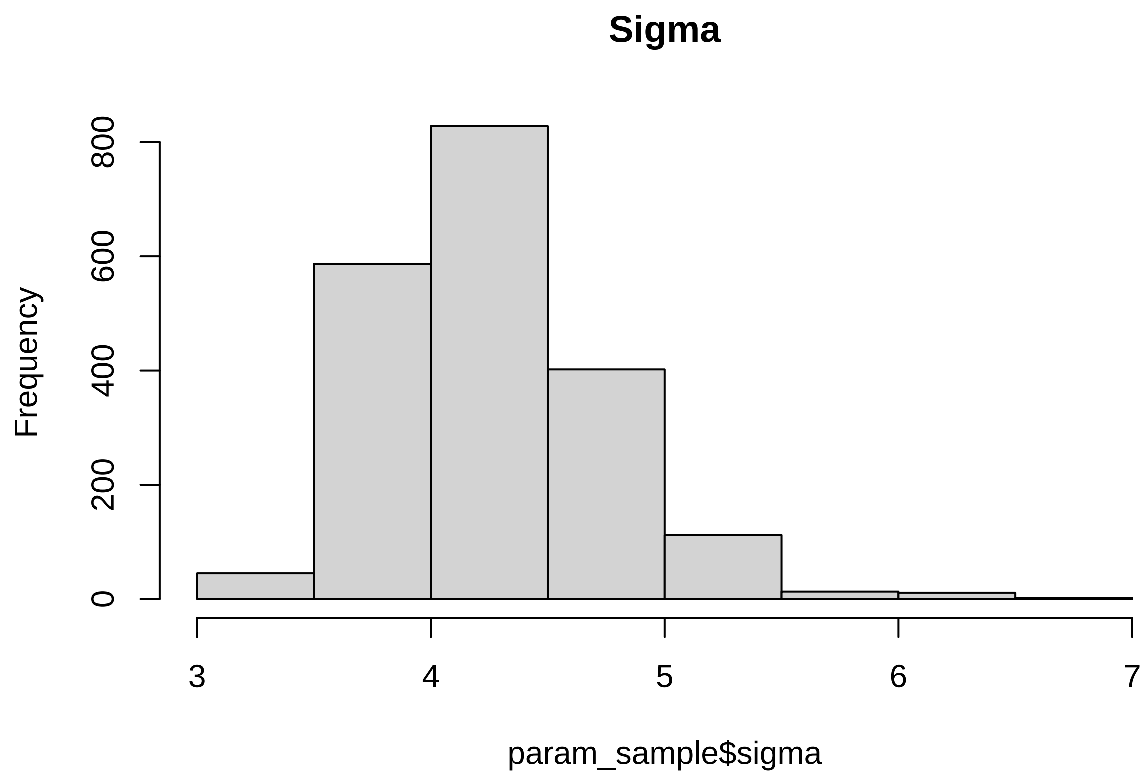
```r
hist(param_sample$beta_0, main = "Beta 0")
```



```r
hist(param_sample$beta_1, main = "Beta 1")
```



```r
hist(param_sample$sigma, main = "Sigma")
```



Si uno se fija apropiadamente en la escala los graficos se parecen bastante.

c.

```r
new <- data.frame(
  displacement = 5)
pp <- posterior_epred(fit_gaussian, newdata = new)
posterior_summary(pp)
```

```
##      Estimate Est.Error     Q2.5    Q97.5
## [1,] 15.94548  1.337263 13.28932 18.49015
```

El posterior predictive coincide.