lezione 24-11-08

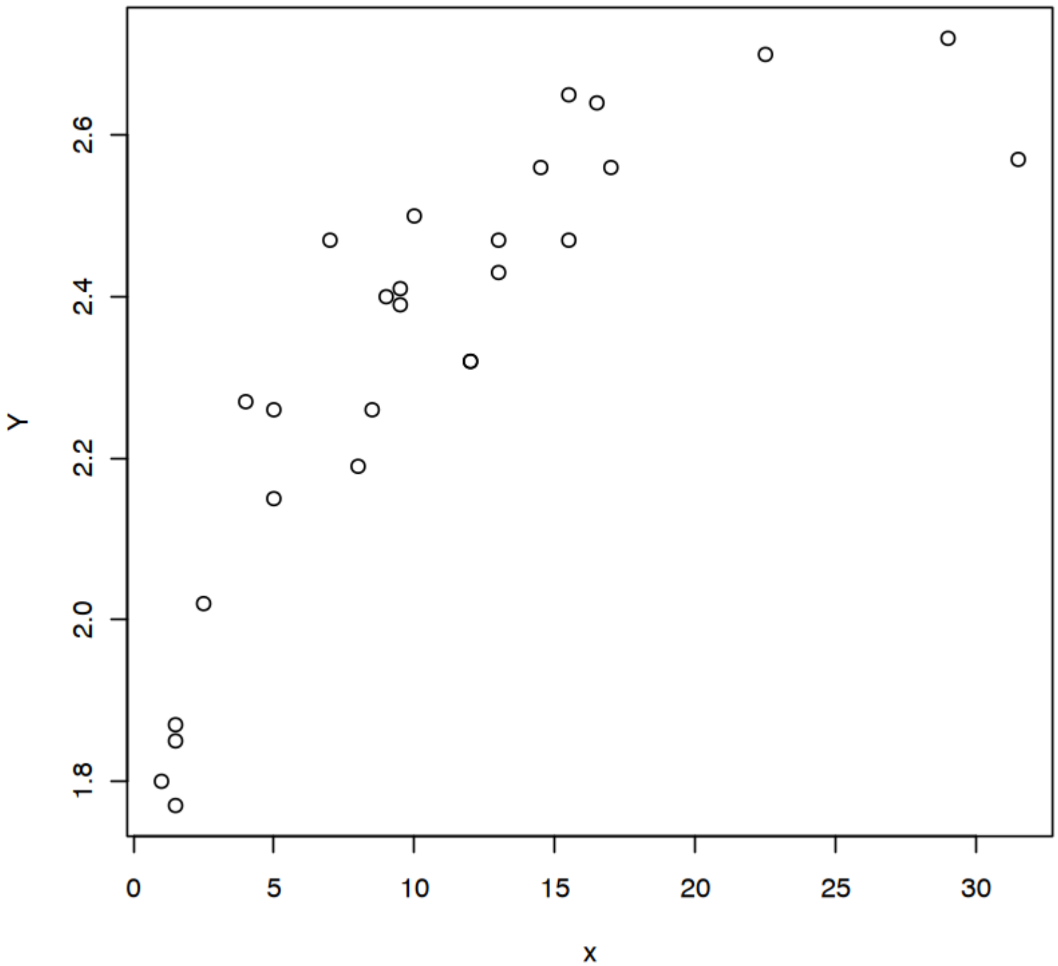
November 8, 2024

[ ]:

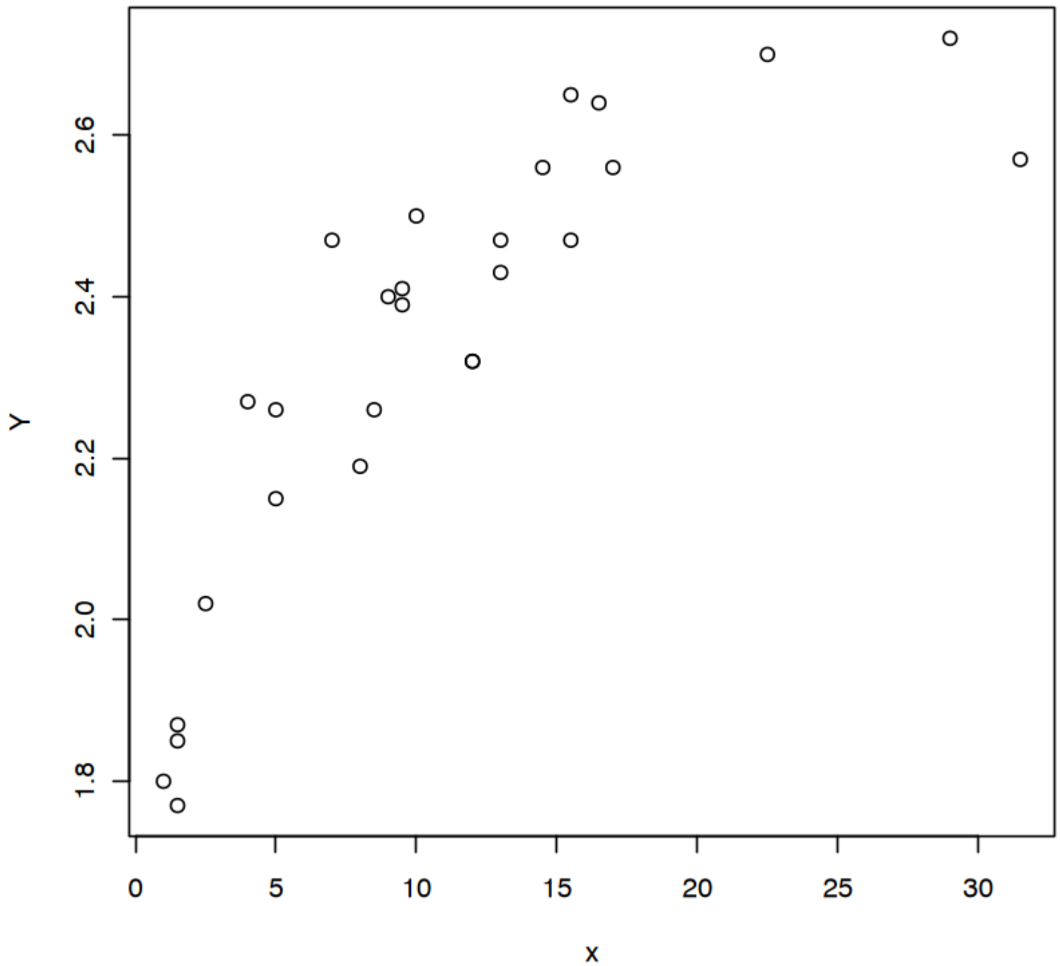
```

```

[ ]:

```r
library(tidyverse)
library(ggplot2)
library(magrittr)
library(MCMCpack)

# valori x
set.seed(1234)
x <- c(
  1.0, 1.5, 1.5, 1.5, 2.5, 4.0, 5.0, 5.0, 7.0,
  8.0, 8.5, 9.0, 9.5, 9.5, 10.0, 12.0, 12.0, 13.0,
  13.0, 14.5, 15.5, 15.5, 16.5, 17.0, 22.5, 29.0, 31.5
)
# valori y
Y <- c(
  1.80, 1.85, 1.87, 1.77, 2.02, 2.27, 2.15, 2.26, 2.47,
  2.19, 2.26, 2.40, 2.39, 2.41, 2.50, 2.32, 2.32, 2.43,
  2.47, 2.56, 2.65, 2.47, 2.64, 2.56, 2.70, 2.72, 2.57
)
```

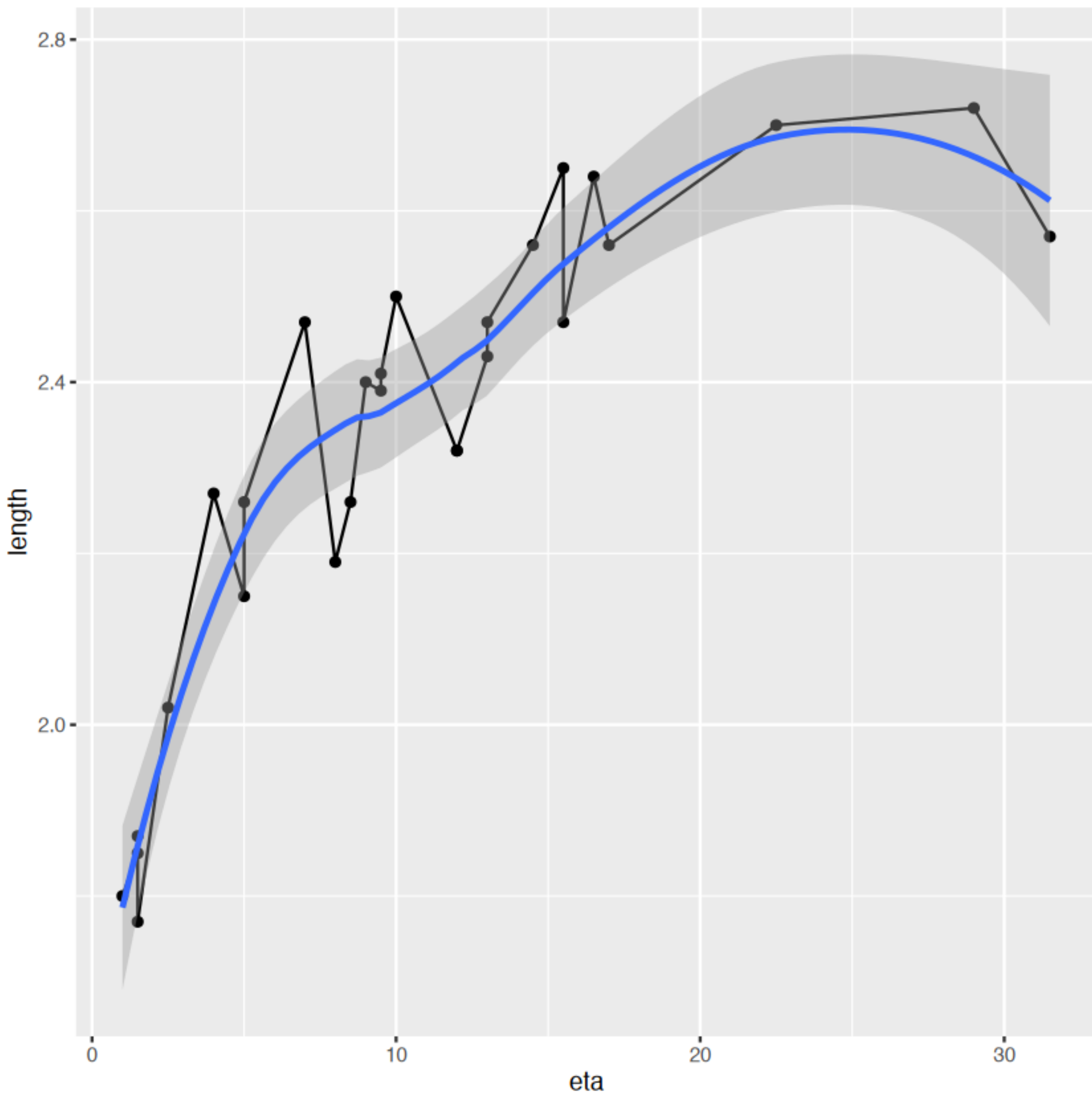


[3]:
```
plot(x, Y)
```



[12]:
```
data_plot <- data.frame(length=Y, eta = x)

p1 <- ggplot(data_plot, aes(x = eta, y = length)) + geom_point()  + geom_line()
 ↪+ geom_smooth()
p1
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



[13]:
```
library(MASS) # per la distribuzione multivariata normale



bayesian_regression <- function(Y, X, tau2 = 10, a = 2, b = 1, n_iter = 10000) {
  # Inizializzazione
  beta_samples <- matrix(NA, n_iter, ncol(X)) # Per salvare i campioni di beta
  sigma2_samples <- numeric(n_iter) # Per salvare i campioni di sigma^2

  # Valori iniziali
  beta <- rep(0, ncol(X))
  sigma2 <- 1

  # MCMC
```

```
  for (i in 1:n_iter) {
    # Aggiorna beta condizionatamente a sigma^2 e Y
    V_beta <- solve(t(X) %*% X / sigma2 + diag(1 / tau2, ncol(X)))
    m_beta <- V_beta %*% t(X) %*% Y / sigma2
    beta <- mvrnorm(1, m_beta, V_beta)

    # Aggiorna sigma^2 condizionatamente a beta e Y
    residuals <- Y - X %*% beta
    shape <- a + length(Y) / 2
    rate <- b + sum(residuals^2) / 2
    sigma2 <- 1 / rgamma(1, shape = shape, rate = rate)

    # Salva i campioni
    beta_samples[i, ] <- beta
    sigma2_samples[i] <- sigma2
  }

  # Risultati
  list(
    beta_samples = beta_samples,
    sigma2_samples = sigma2_samples
  )
}
```

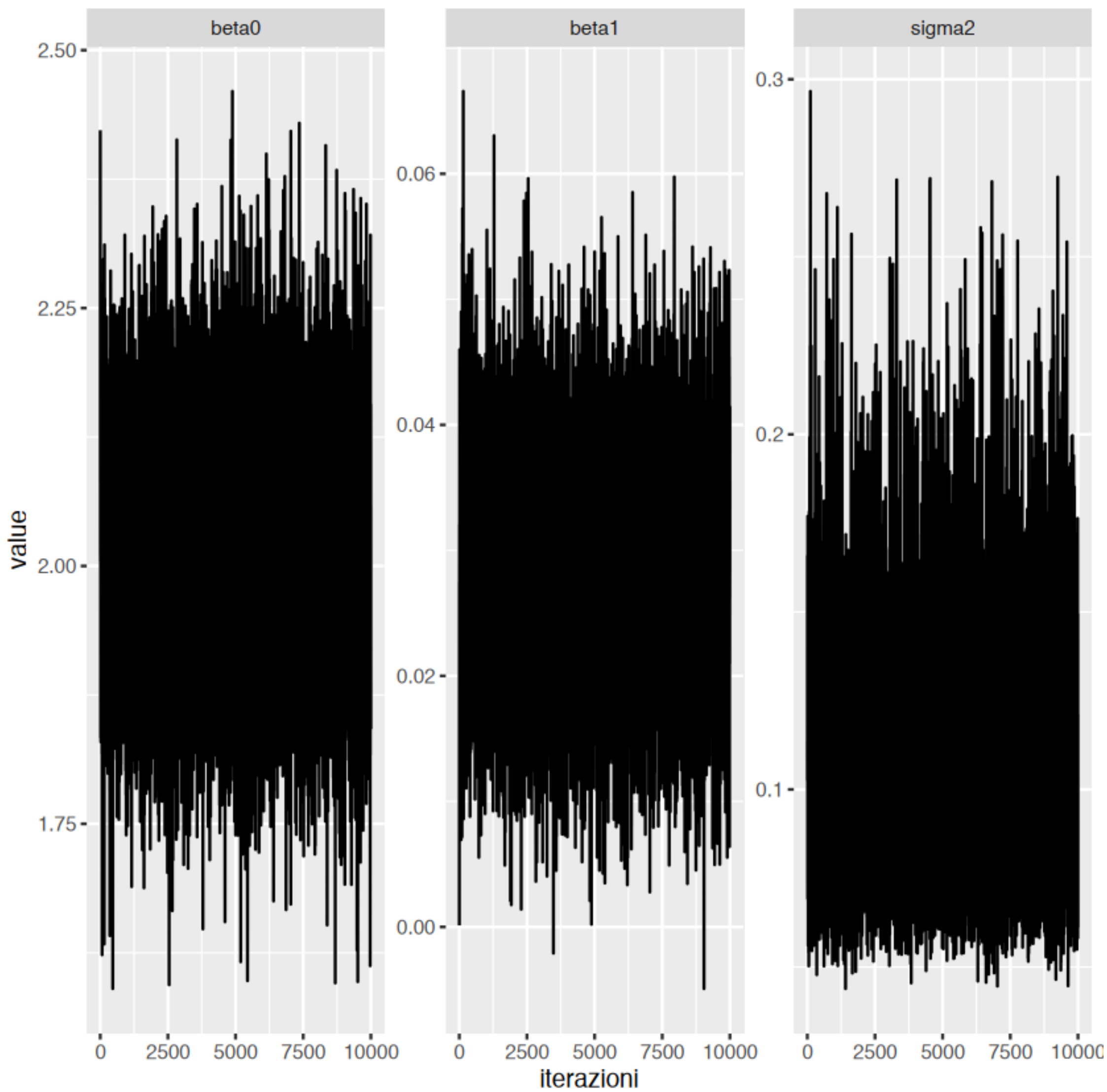
[15]:
```
n <- length(Y)
M1_post_samples <- bayesian_regression(Y = Y, X= cbind(rep(1, n), x), tau2 =
  ↪1000000, a = 1, b = 1, n_iter = 10000)
```

[26]:
```
data_plot_mcmc <- data.frame(
  "iterazioni" = 1:nrow(M1_post_samples$beta_samples),
  "beta0" = M1_post_samples$beta_samples[,1],
  "beta1" = M1_post_samples$beta_samples[,2],
  "sigma2" = M1_post_samples$sigma2_samples)

data_plot_long <- data_plot_mcmc %>% pivot_longer(
  cols = colnames(data_plot_mcmc)[-1],
  names_to = "parameters",
  values_to = "value"

)
data_plot_long[1:10,]
```

A tibble: 10 x 3

iterazioni <int>	parameters <chr>	value <dbl>
1	beta0	2.4224523467
1	beta1	0.0001230277
1	sigma2	0.1135394259
2	beta0	2.2834821927
2	beta1	0.0105244138
2	sigma2	0.1027392490
3	beta0	2.0814166126
3	beta1	0.0274681521
3	sigma2	0.1127173166
4	beta0	2.1019042141

[31]:
```
data_plot_long %>% ggplot(aes(x = iterazioni, y = value)) +
  geom_line() + facet_wrap(~parameters, scales = "free_y")
```



[ ]:

```

```

$\mu^b = \beta_0^b + \beta_1^b x$

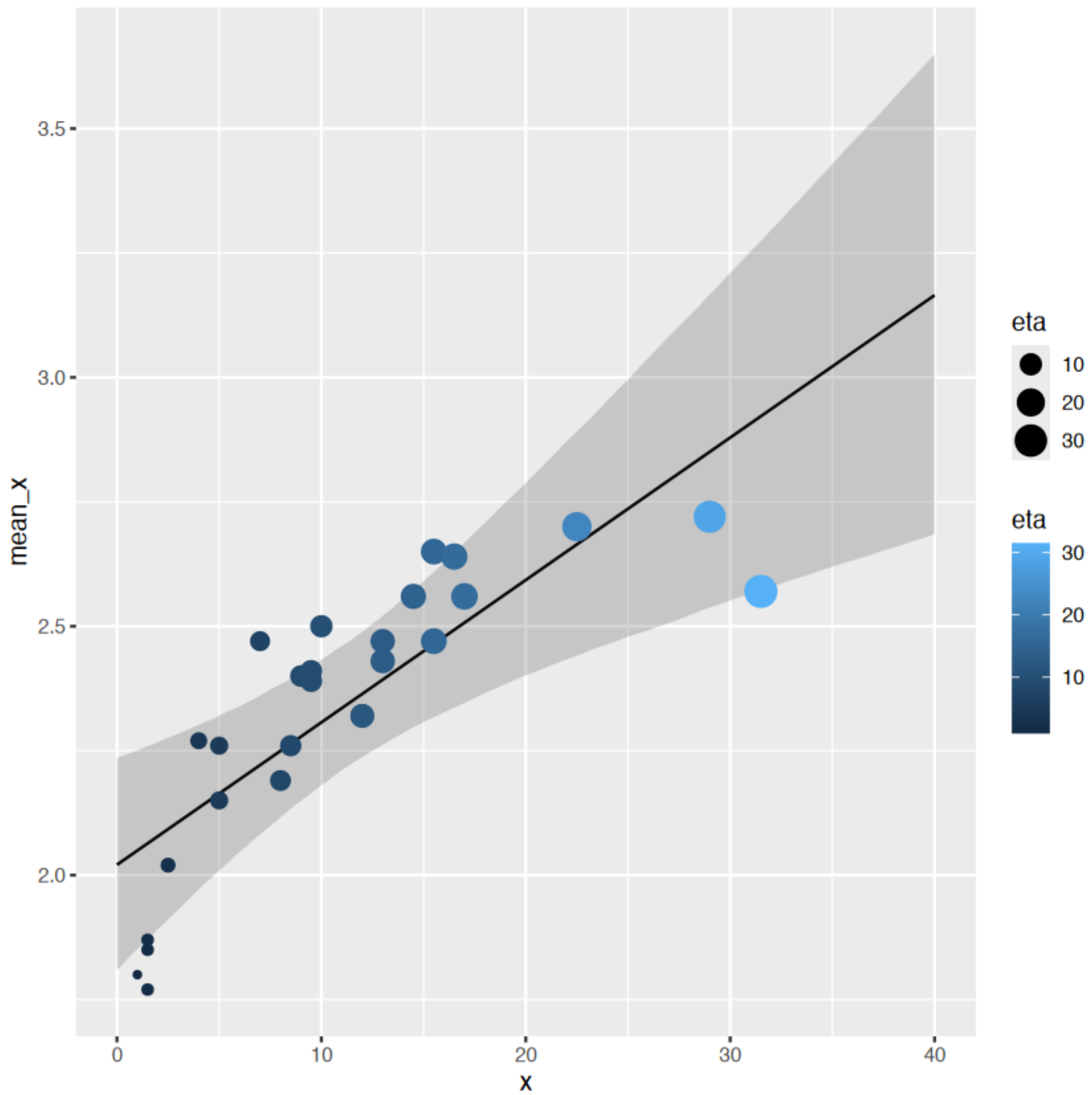
$f(\mu|\mathbf{y})$

[35]:

```
age_vec <- seq(0,40, by  = 0.1)
mean_vec <- rep(0, length(age_vec))
q1 <- rep(0, length(age_vec))
q2 <- rep(0, length(age_vec))

for(ix in 1:length(age_vec))
{
  sim_post <- M1_post_samples$beta_samples[, 1] + M1_post_samples$beta_samples[, 2] * age_vec[ix]
  mean_vec[ix] <- mean(sim_post)
  q1[ix] <- quantile(sim_post, probs = 0.025)
  q2[ix] <- quantile(sim_post, probs = 1-0.025)
}

data_plot_mean <- data.frame(x = age_vec, mean_x = mean_vec, q1 = q1 , q2 = q2)
```

[43]:

```
data_obs <- data.frame(length = Y, eta = x)
data_plot_mean %>% ggplot(aes(x = x, y = mean_x)) +
  geom_line() +
  geom_ribbon(aes(ymin = q1, ymax = q2), alpha = 0.2) +
  geom_point(data = data_obs, aes(y = length, x = eta ,size = eta, col= eta))
```



[44]:
```
bayesian_mcmc_with_gamma <- function(Y, X, tau2 = 10, a = 2, b = 1, n_iter =
↪10000, a_gamma = 0.5, b_gamma = 1) {
  # Number of data points
  n <- length(Y)
  # Initialize vectors for samples
  beta_0_samples <- numeric(n_iter)
  beta_1_samples <- numeric(n_iter)
  sigma2_samples <- numeric(n_iter)
  gamma_samples <- numeric(n_iter)

  # Initial values
  beta_0 <- 3
  beta_1 <- 1
```

```r
sigma2 <- 1
gamma <- runif(1, a_gamma, b_gamma)

# MCMC sampling
for (i in 1:n_iter) {

  # Update beta_0 (conditional on beta_1, gamma, x, Y, sigma^2)
  V_beta_0 <- solve(n / sigma2 + 1 / tau2)
  m_beta_0 <- V_beta_0 %*% (sum(Y + beta_1 * gamma^X) / sigma2)
  beta_0 <- mvrnorm(1, m_beta_0, V_beta_0)

  # Update beta_1 (conditional on beta_0, gamma, x, Y, sigma^2)
  V_beta_1 <- solve(sum((gamma^X)^2) / sigma2 + 1 / tau2)
  m_beta_1 <- V_beta_1 %*% sum(gamma^X * (-Y + beta_0)) / sigma2
  beta_1 <- mvrnorm(1, m_beta_1, V_beta_1)

  # Update sigma^2 (Inverse Gamma prior)
  residuals <- Y - (beta_0 - beta_1 * gamma^X)
  shape <- a + n / 2
  rate <- b + sum(residuals^2) / 2
  sigma2 <- rinvgamma(1, shape, rate)

  # Update gamma (using Metropolis-Hastings)
  gamma_proposal <- runif(1, max(gamma - 0.1, a_gamma), min(gamma + 0.1, b_gamma)) # distribuzione Q
  residuals_proposal <- Y - (beta_0 - beta_1 * gamma_proposal^X)
  log_likelihood_proposal <- -0.5 * sum((residuals_proposal)^2) / sigma2
  residuals_current <- Y - (beta_0 - beta_1 * gamma^X)
  log_likelihood_current <- -0.5 * sum((residuals_current)^2) / sigma2

  log_q_proposal <- log(1 / (min(gamma + 0.1, b_gamma) - max(gamma - 0.1, a_gamma)))
  log_q_current <- log(1 / (min(gamma_proposal + 0.1, b_gamma) - max(gamma_proposal - 0.1, a_gamma)))

  log_ratio_numeratore <- log_likelihood_proposal + log_q_current
  log_ratio_denominatore <- log_likelihood_current + log_q_proposal

  if (runif(1, 0, 1) < exp(log_ratio_numeratore - log_ratio_denominatore)) {
    gamma <- gamma_proposal
  }

  # Save the samples
  beta_0_samples[i] <- beta_0
  beta_1_samples[i] <- beta_1
  sigma2_samples[i] <- sigma2
  gamma_samples[i] <- gamma
```

```
  }

  # Return the results
  list(
    beta_0_samples = beta_0_samples,
    beta_1_samples = beta_1_samples,
    sigma2_samples = sigma2_samples,
    gamma_samples = gamma_samples
  )
}
```

[45]:
```
M1_v2_post <- bayesian_mcmc_with_gamma(Y=Y, X=x, tau2 = 100000, a = 1, b = 1,␣
 ↪n_iter = 10000, a_gamma = 0.5, b_gamma = 1)
```

[54]:
```
str(M1_v2_post)
data_plot_mcmc <- data.frame(
  "iterazioni" = 1:length(M1_v2_post$beta_0_samples),
  "beta0" = M1_v2_post$beta_0_samples,
  "beta1" = M1_v2_post$beta_1_samples,
  "sigma2" = M1_v2_post$sigma2_samples,
  "gamma" = M1_v2_post$gamma_samples
)
data_plot_mcmc[1:10,]
data_plot_long <- data_plot_mcmc %>% pivot_longer(
  cols = colnames(data_plot_mcmc)[-1],
  names_to = "parameters",
  values_to = "value"
)
```

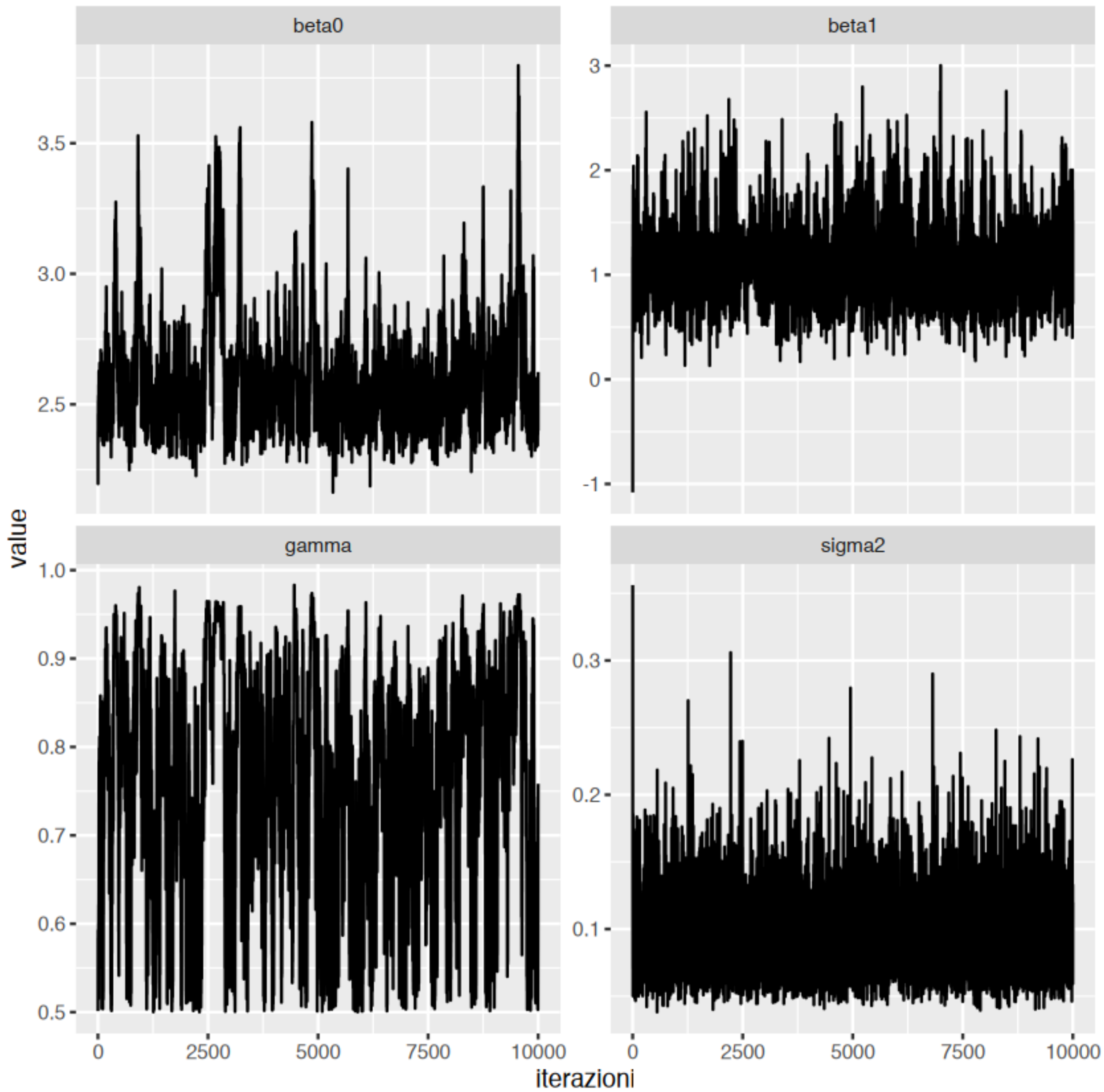
```
List of 4
 $ beta_0_samples: num [1:10000] 2.4 2.2 2.46 2.35 2.44 …
 $ beta_1_samples: num [1:10000] -1.081 1.167 0.356 1.143 1.269 …
 $ sigma2_samples: num [1:10000] 0.3561 0.1397 0.109 0.0709 0.1241 …
 $ gamma_samples : num [1:10000] 0.501 0.584 0.553 0.593 0.546 …
```

A data.frame: 10 x 5

	iterazioni <int>	beta0 <dbl>	beta1 <dbl>	sigma2 <dbl>	gamma <dbl>
1	1	2.397750	-1.0806661	0.35614665	0.5012504
2	2	2.195909	1.1666644	0.13970359	0.5839929
3	3	2.462303	0.3560844	0.10895024	0.5530770
4	4	2.351409	1.1434140	0.07092931	0.5925998
5	5	2.436060	1.2686638	0.12413472	0.5464091
6	6	2.449148	0.9425077	0.05015845	0.5900977
7	7	2.513276	1.4827295	0.18574117	0.5393094
8	8	2.392960	0.5414237	0.09407444	0.5640607
9	9	2.391524	1.6131458	0.12303842	0.5456122
10	10	2.532764	1.7893774	0.10731064	0.5456122

[55]:

```r
data_plot_long %>% ggplot(aes(x = iterazioni, y = value)) +
  geom_line() +
  facet_wrap(~parameters, scales = "free_y")
```



[ ]:

```r
age_vec <- seq(0, 40, by = 0.1)
mean_vec <- rep(0, length(age_vec))
q1 <- rep(0, length(age_vec))
q2 <- rep(0, length(age_vec))

for (ix in 1:length(age_vec))
{
  sim_post <- M1_v2_post$beta_0_samples - M1_v2_post$beta_1_samples *
 ↪M1_v2_post$gamma_samples^age_vec[ix]
```

```
  mean_vec[ix] <- mean(sim_post)
  q1[ix] <- quantile(sim_post, probs = 0.025)
  q2[ix] <- quantile(sim_post, probs = 1 - 0.025)
}

data_plot_mean <- data.frame(x = age_vec, mean_x = mean_vec, q1 = q1, q2 = q2)
```

[57]:
```
data_obs <- data.frame(length = Y, eta = x)
data_plot_mean %>% ggplot(aes(x = x, y = mean_x)) +
  geom_line() +
  geom_ribbon(aes(ymin = q1, ymax = q2), alpha = 0.2) +
  geom_point(data = data_obs, aes(y = length, x = eta, size = eta, col = eta))
```