Exercise 9 - MCMC - Gibbs Sampling, Metropolis-Hastings Sampling

Stefan Merdian

2025-01-12

```
rho <- 0.5
M <- 30000
```

a) Implement a Gibbs sampler

```
gibbs_sampler <- function(rho, M) {
  samples <- matrix(0, nrow = M, ncol = 2)
  colnames(samples) <- c("theta1", "theta2")

  theta1 <- 0
  theta2 <- 0

  for (i in 1:M) {
    mu1 <- rho * theta2
    sigma1 <- sqrt(1 - rho^2)
    theta1 <- rnorm(1, mean = mu1, sd = sigma1)

    mu2 <- rho * theta1
    sigma2 <- sqrt(1 - rho^2)
    theta2 <- rnorm(1, mean = mu2, sd = sigma2)

    samples[i, ] <- c(theta1, theta2)
  }

  return(samples)
}
```

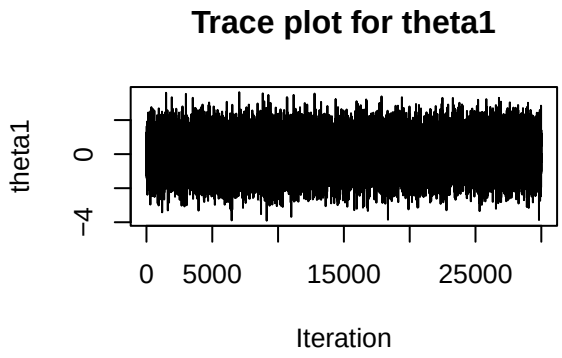
We use the Gibbs sampler to generate samples from a bivariate normal distribution with parameters and M=30000. Than we will show the sampled values of 1 and 2 over iterations.

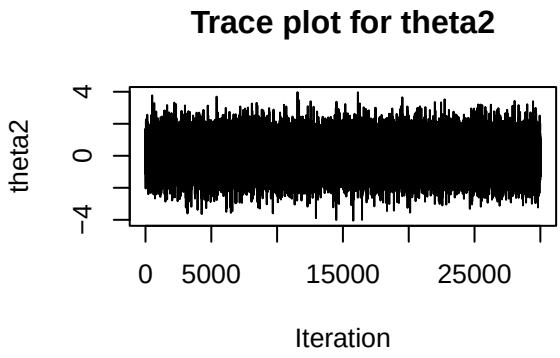
Scatter plot: Displays the relationship between 1 and 2

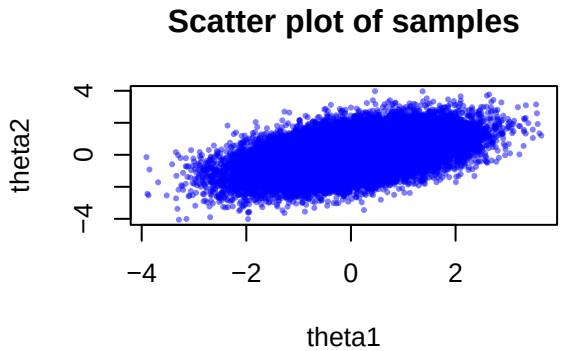
```
set.seed(123)
gibbs_samples <- gibbs_sampler(rho, M)

par(mfrow = c(2, 2))

plot(gibbs_samples[, 1], type = "l", main = "Trace plot for theta1", xlab = "Iteration", ylab = "theta1
```

```
plot(gibbs_samples[, 2], type = "l", main = "Trace plot for theta2", xlab = "Iteration", ylab = "theta2
```

```
plot(gibbs_samples, main = "Scatter plot of samples", xlab = "theta1", ylab = "theta2", pch = 16, cex =
```

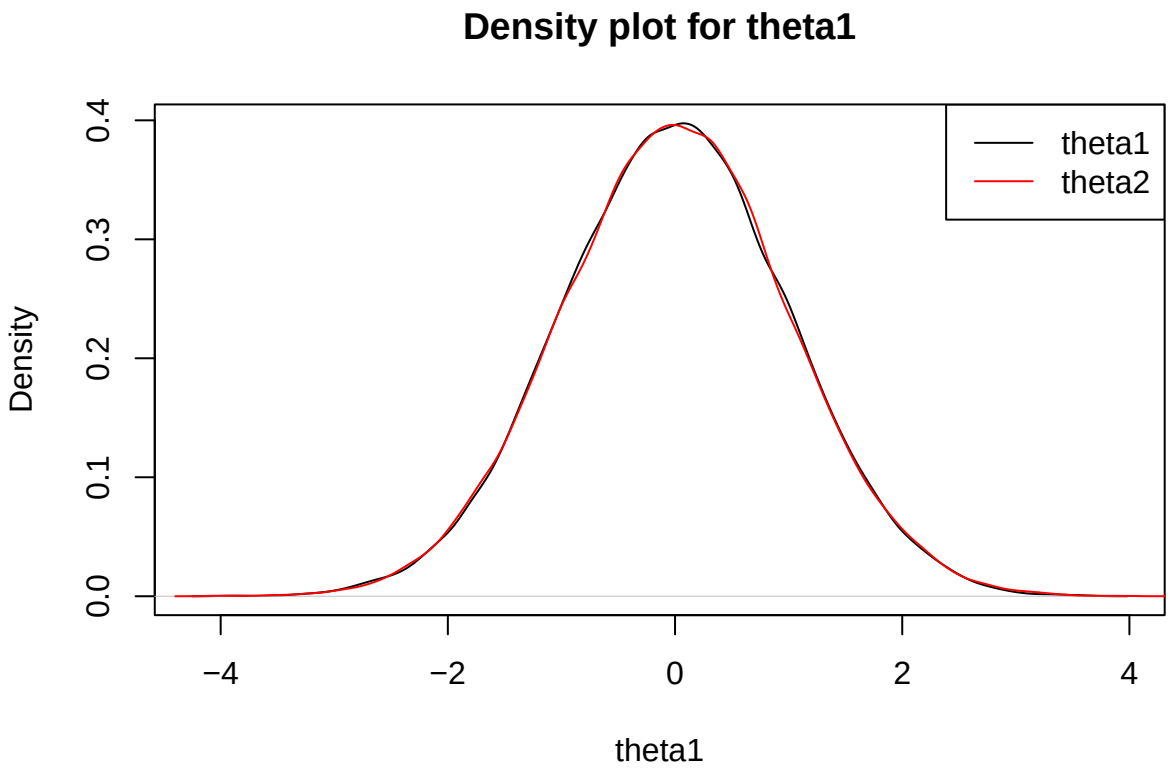






We also estimate the probability density functions of 1 and 2 from the Gibbs samples.

```
density_theta1 <- density(gibbs_samples[, 1])
plot(density_theta1, main = "Density plot for theta1", xlab = "theta1", ylab = "Density")
density_theta2 <- density(gibbs_samples[, 2])
lines(density_theta2, col = "red")
legend("topright", legend = c("theta1", "theta2"), col = c("black", "red"), lty = 1)
```




```
# Print summary of the samples
summary(gibbs_samples)
```

```
##      theta1              theta2
##  Min.   :-3.908289   Min.   :-4.055195
##  1st Qu.:-0.678665   1st Qu.:-0.672311
##  Median : 0.004087   Median : 0.004173
##  Mean   : 0.000094   Mean   : 0.001358
##  3rd Qu.: 0.669110   3rd Qu.: 0.669916
##  Max.   : 3.638826   Max.   : 3.973645
```

b) Metropolis–Hastings algorithm with block-wise update

1. Initialization: Start with an initial value for =(0,0)
2. Target density: Define the bivariate normal target density function
3. Proposal step: Propose a new value for 1 or 2 using a normal distribution centered at the current value with a standard deviation (proposal_sd = 1).
4. Acceptance step: Calculate the acceptance probability for the proposed value based on the ratio of target densities. Accept or reject the proposed value based on a random draw.
5. Repeat: Update iteratively for MM iterations, storing the samples at each step.

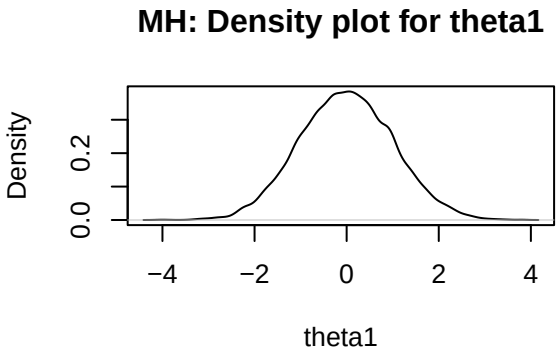
```
mh_sampler <- function(rho, M) {
  samples <- matrix(0, nrow = M, ncol = 2)
  colnames(samples) <- c("theta1", "theta2")

  theta <- c(0, 0)
```

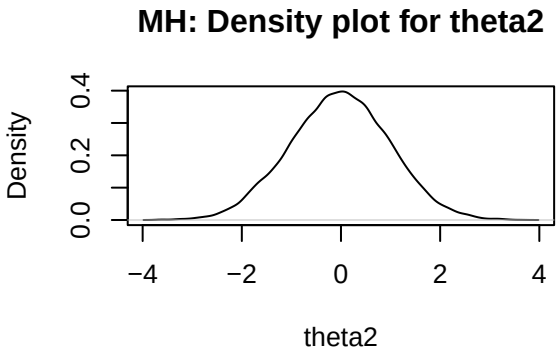
```
  target_density <- function(theta) {
    exp(-0.5 / (1 - rho^2) * (theta[1]^2 - 2 * rho * theta[1] * theta[2] + theta[2]^2))
  }

  proposal_sd <- 1

  for (i in 1:M) {
    for (j in 1:2) {
      theta_proposed <- theta
      theta_proposed[j] <- rnorm(1, mean = theta[j], sd = proposal_sd)

      acceptance_prob <- min(1, target_density(theta_proposed) / target_density(theta))

      if (runif(1) < acceptance_prob) {
        theta <- theta_proposed
      }
    }

    samples[i, ] <- theta
  }

  return(samples)
}
```

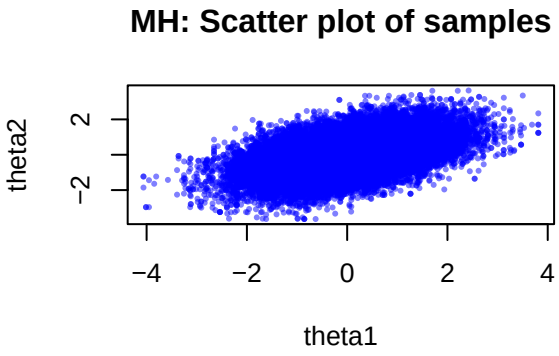
```
mh_samples <- mh_sampler(rho, M)
```

We also estimate the probability density functions of 1 and 2 from the MH algorithm.

```
par(mfrow = c(2, 2))
plot(density(mh_samples[, 1]), main = "MH: Density plot for theta1", xlab = "theta1", ylab = "Density")
plot(density(mh_samples[, 2]), main = "MH: Density plot for theta2", xlab = "theta2", ylab = "Density")

# Scatter plot
plot(mh_samples, main = "MH: Scatter plot of samples", xlab = "theta1", ylab = "theta2", pch = 16, cex =
```








```
cat("Summary of Metropolis-Hastings samples:\n")
```

```
## Summary of Metropolis-Hastings samples:
```

```
print(summary(mh_samples))
```

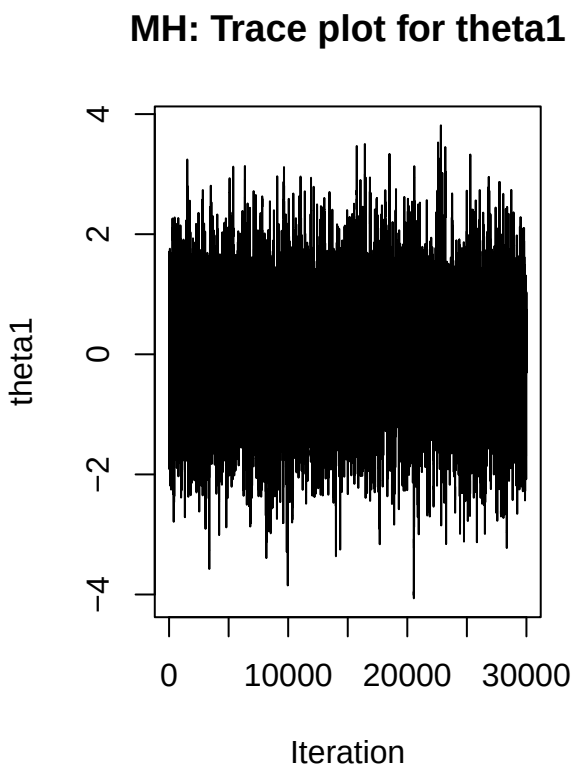
```
##      theta1              theta2
##  Min.   :-4.062732   Min.   :-3.63653
##  1st Qu.:-0.699890   1st Qu.:-0.70048
##  Median :-0.007292   Median :-0.01720
##  Mean   :-0.004950   Mean   :-0.02552
##  3rd Qu.: 0.684021   3rd Qu.: 0.65206
##  Max.   : 3.812306   Max.   : 3.63415
```

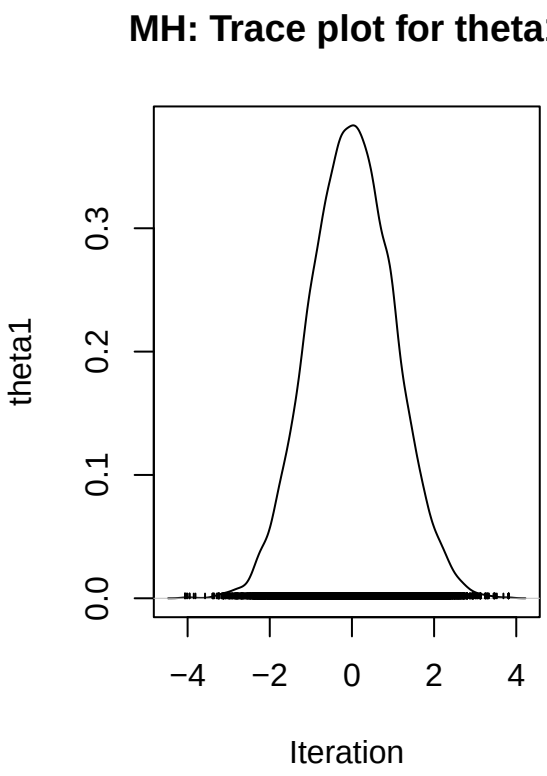
chain diagnostic

```
library(coda)
```

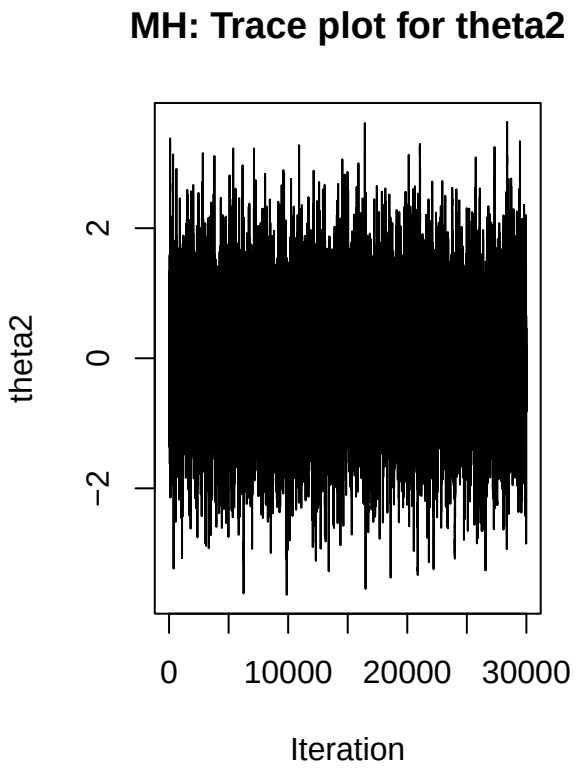
```
## Warning: package 'coda' was built under R version 4.4.2
```

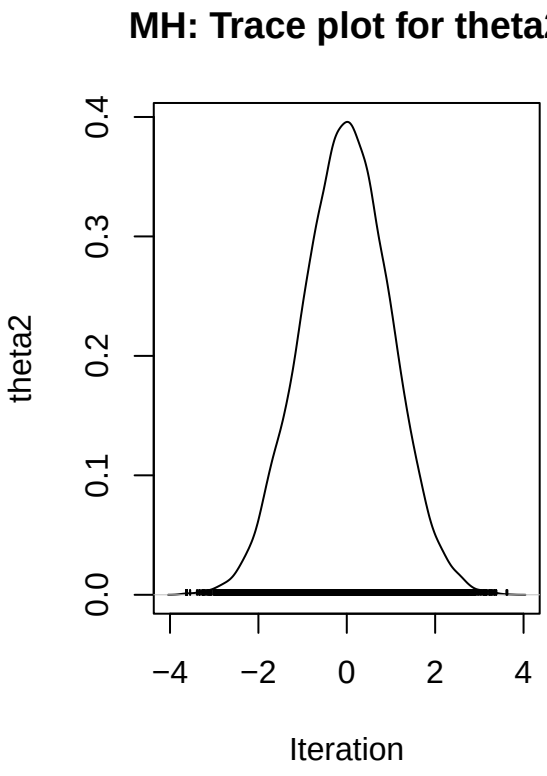
```
gibbs_mcmc <- as.mcmc(gibbs_samples)
mh_mcmc <- as.mcmc(mh_samples)
par(mfrow = c(2, 1))
plot(mh_mcmc[, 1], type = "l", main = "MH: Trace plot for theta1", xlab = "Iteration", ylab = "theta1")
```



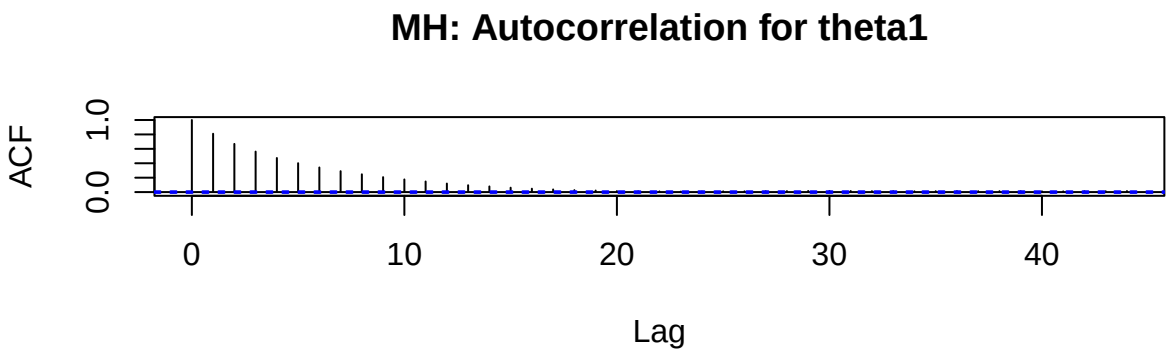


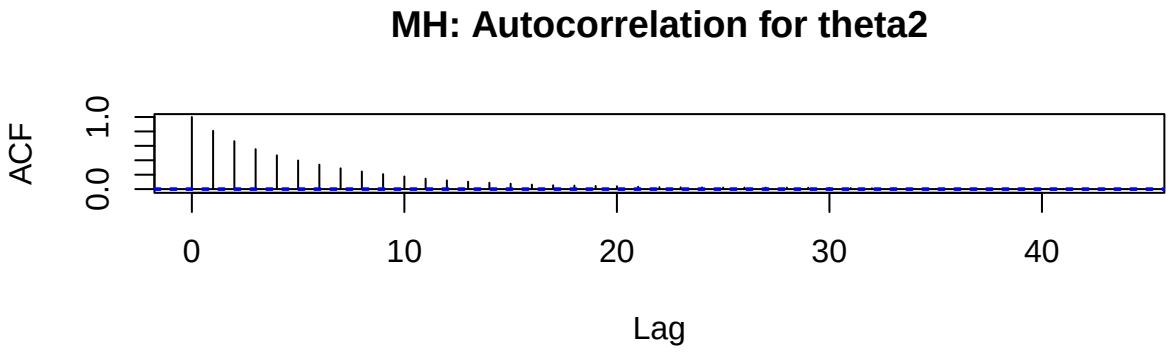

```
plot(mh_mcmc[, 2], type = "l", main = "MH: Trace plot for theta2", xlab = "Iteration", ylab = "theta2")
```






```
acf(mh_mcmc[, 1], main = "MH: Autocorrelation for theta1")
acf(mh_mcmc[, 2], main = "MH: Autocorrelation for theta2")
```





Trace Plots:

- The left plot (in 1/2) for both 1 and 2, show the sampled values over the iterations
- The chains appear well-mixed, with no visible trends or drifts, indicating that the algorithm is exploring the parameter space effectively and has likely converged.

Density plots (right plot in 1/2):

- Also for both the densities are symmetric and approximately centered around 0, consistent with the bivariate normal distribution.

Autocorrelation Plots:

- The autocorrelation decreases rapidly with lag, which is a good sign of efficient mixing.
- Low autocorrelation indicates that successive samples are less dependent on each other, improving the effective sample size.

```
# Effective sample size
cat("Effective sample size (MH):\n")
```

```
## Effective sample size (MH):
```

```
print(effectiveSize(mh_mcmc))
```

```
##   theta1   theta2
## 2748.768 2728.083
```

```r
# Geweke diagnostic
cat("Geweke diagnostic (MH):\n")
```

```
## Geweke diagnostic (MH):
```

```r
print(geweke.diag(mh_mcmc))
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## theta1 theta2
## 0.3385 0.8182
```

Effective Sample Size:

- The effective sample size indicates the number of independent samples the chain effectively provides.
- While the chain has 30,000 iterations, the effective sample size is smaller due to autocorrelation. However, these values are high, showing the chain mixes well.

Geweke Diagnostic:

- These values represent z-scores. Values near 0 indicate no significant difference between the two segments, suggesting convergence.
- Both 1 and 2 have values close to 0, confirming the chains have likely converged.