STA 602 Lab 6

Yicheng Shen

24 October, 2022

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \sim N(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.9 & 0.1 \\ 0.9 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{bmatrix})$$

Exercise 1

$$\begin{aligned} \mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} &\sim N(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}) \\ U_1|U_2 &= N(\mu, \Sigma) \\ \mu &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(U_2 - \mu_2) \\ \Sigma &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned}$$

Exercise 2

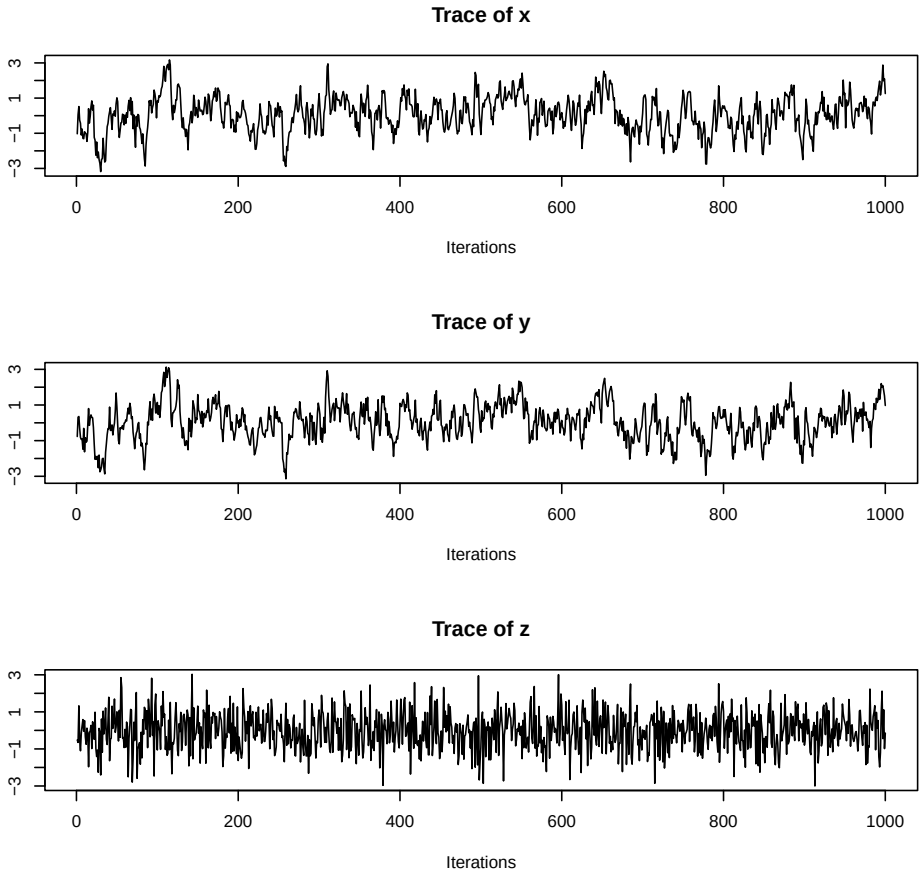
The three conditional distributions are specified below in the Gibbs sampler (using the formula above).

Exercise 3

The mixing for x and y draws are terrible. Their effective sample sizes are low as expected. We see very high, slowly decaying ACF as well.

```
# initialization
x=y=z=0
# result
S = 1000
POST = matrix(NA, S, 3)
# gibbs sampling
for(i in 1:S){
# update x
x = rnorm(1,
          mean = 0 + matrix(c(0.9, 0.1), 1, 2) %*% solve(matrix(c(1, 0.1, 0.1, 1), 2, 2)) %*% matrix(c(y
          sd = sqrt(1 - matrix(c(0.9, 0.1), 1, 2) %*% solve(matrix(c(1, 0.1, 0.1, 1), 2, 2)) %*% matrix
          )
# update y
y = rnorm(1,
          mean = 0 + matrix(c(0.9, 0.1), 1, 2) %*% solve(matrix(c(1, 0.1, 0.1, 1), 2, 2)) %*% matrix(c(x
```

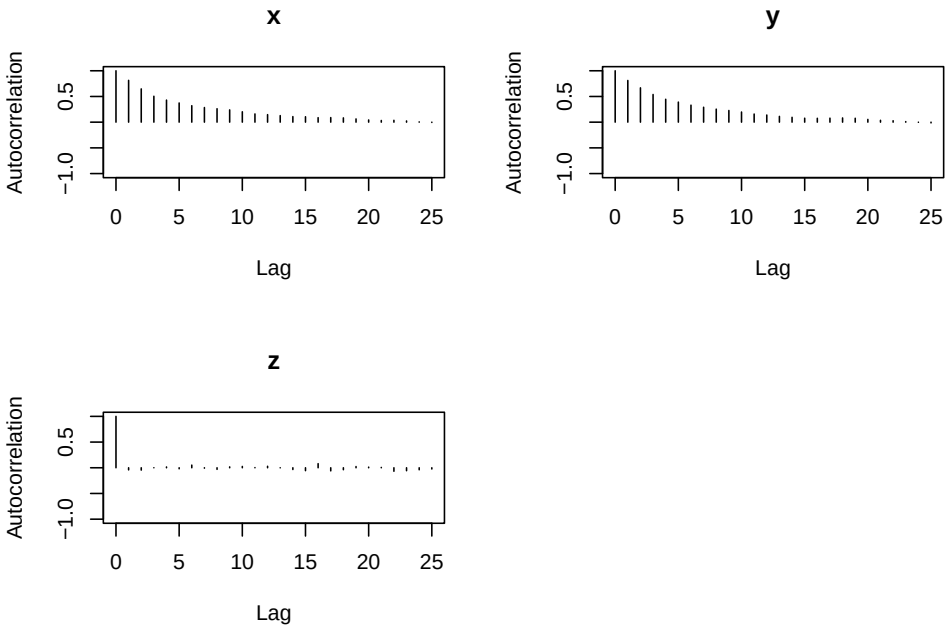
```
          sd = sqrt(1 - matrix(c(0.9, 0.1), 1, 2) %*% solve(matrix(c(1, 0.1, 0.1, 1), 2, 2)) %*% matrix
          )
# update z
z = rnorm(1,
          mean = 0 + matrix(c(0.1, 0.1), 1, 2) %*% solve(matrix(c(1, 0.9, 0.9, 1), 2, 2)) %*% matrix(c(
          sd = sqrt(1 - matrix(c(0.1, 0.1), 1, 2) %*% solve(matrix(c(1, 0.9, 0.9, 1), 2, 2)) %*% matrix
          )
# save
  POST[i,] = c(x, y, z)
}

colnames(POST) = c("x", "y", "z")
coda_obj = mcmc(POST)
par(mfrow = c(3,1))
# trace plot
traceplot(coda_obj)
```




```
effectiveSize(coda_obj)
```

```
##          x          y          z
##   94.44445  104.09546 1000.00000
```

```
# autocorrelation plots
autocorr.plot(coda_obj)
```



Exercise 4

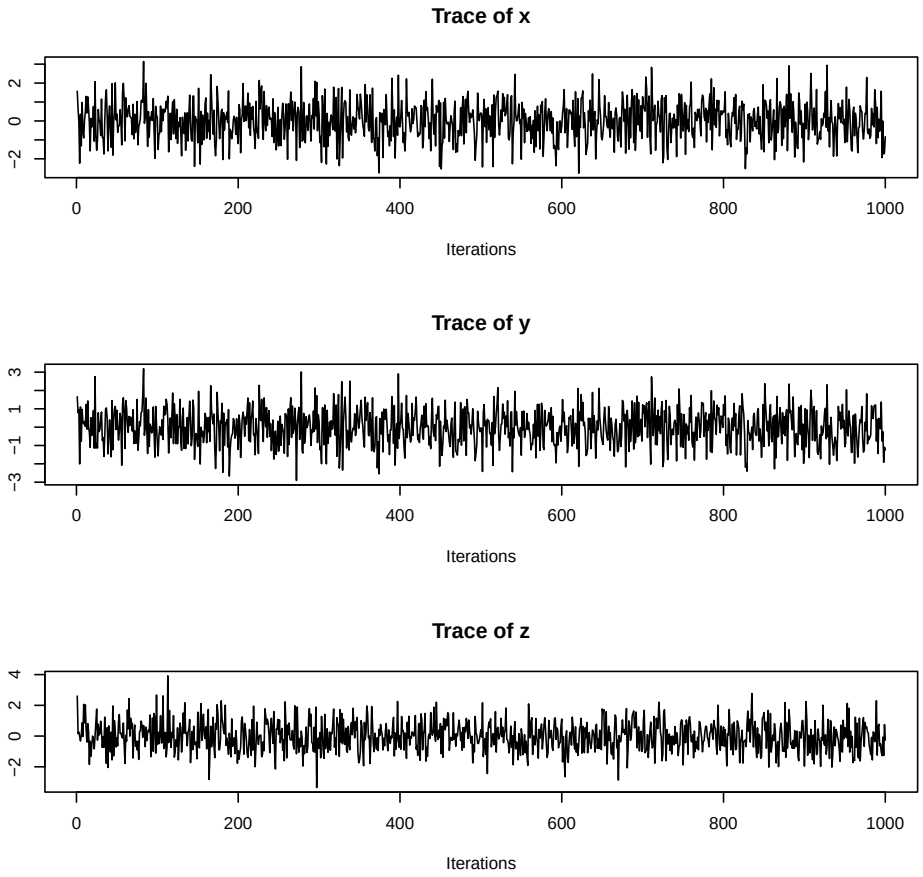
The blocked conditionals are shown below.

Exercise 5

The mixing is pretty good in this case.

```
# initialization
x=y=z=0
# result
S = 1000
POST = matrix(NA, S, 3)
# gibbs sampling
for(i in 1:S){
# update x y
  xy = mvtnorm::rmvnorm(1,
    c(0, 0) + matrix(c(0.1, 0.1), 2, 1) %*% 1 %*% matrix(z) ,
    (matrix(c(1, 0.9, 0.9, 1), 2, 2) - matrix(c(0.1, 0.1), 2, 1) %*% 1 %*% matrix(c(0.1, 0.1), 1, 2))
  )
  # update z
  z = rnorm(1,
    mean = 0 +
      matrix(c(0.1, 0.1), 1, 2) %*% solve(matrix(c(1, 0.9, 0.9, 1), 2, 2)) %*% matrix(c(x, y)) ,
    sd = sqrt(1 -
                matrix(c(0.1, 0.1), 1, 2) %*% solve(matrix(c(1, 0.9, 0.9, 1), 2, 2)) %*% matrix(c(0.1,
```

```
  )

# save
  POST[i,] = c(xy, z)
}

colnames(POST) = c("x", "y", "z")
coda_obj = mcmc(POST)
par(mfrow = c(3,1))
# trace plot
traceplot(coda_obj)
```




```
effectiveSize(coda_obj)
```

```
##    x    y    z
## 1000 1000 1000
```

Exercise 6

The first is not efficient due to the high dependence between x and y. While in 2nd case, (x,y) and z are less dependent on each other.