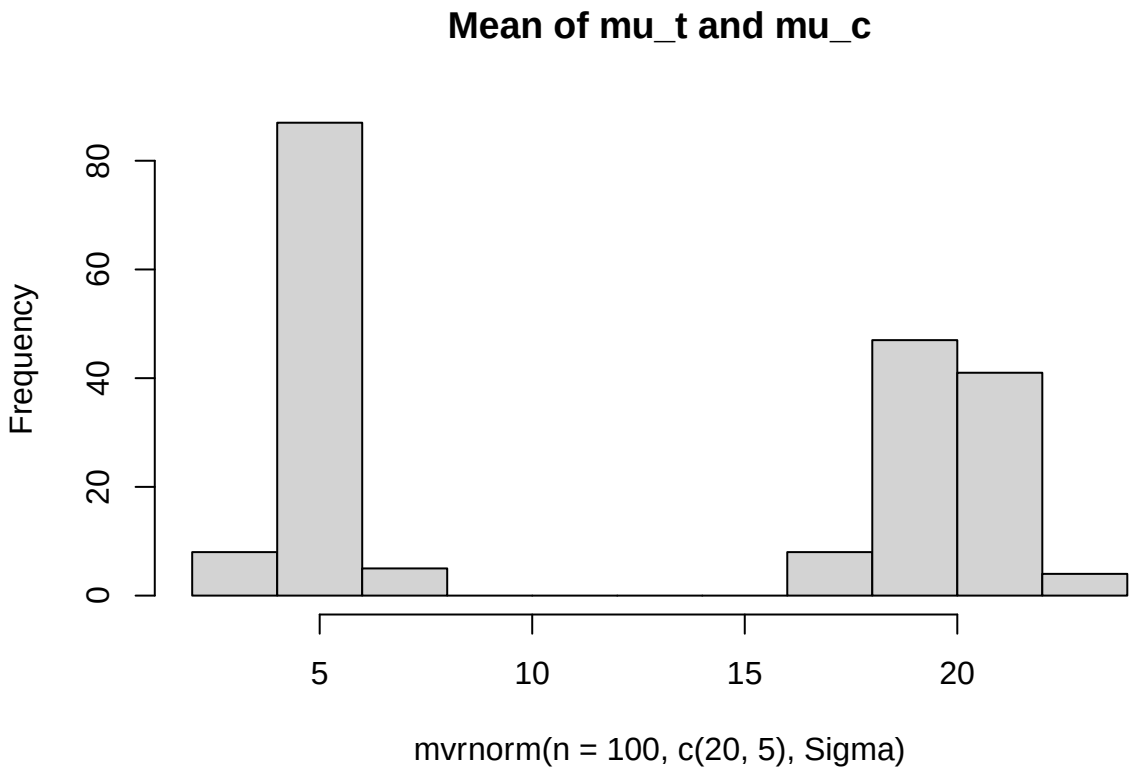
Problem 15.1

15.1.1

Use your statistical software of choice generate 100 independent samples of (t, c). Draw a scatter plot of your (t, c) samples, with lines connecting consecutive points. How close are the sample-estimated means to the true means? (Hint: to do this in R you will need to use the MASS package:

```
library (MASS)
Sigma <- matrix(c(2,0.8,0.8,0.5),2,2)
hist(mvrnorm(n = 100, c(20,5), Sigma), main = "Mean of mu_t and mu_c")
```



15.1.2

Code up a Random Walk Metropolis sampler for this example. This is composed of the following steps:

```
ldebug <- 0
Sigma <- matrix(c(2, 0.8, 0.8, 0.5), 2, 2)
```

1

Create a proposal function that takes a current value of = (t, c) and outputs a proposed value of these using a multivariate normal centred on the current estimates. (Here use a multivariate normal proposal with an identity covariance matrix.)

```
fproposal <- function(theta_current){
  if (ldebug == 1) cat("\nfproposal:theta_current = ", theta_current)
  return(mvrnorm(n=1, mu = theta_current,Sigma = Sigma))
}
```

2

Create a function which takes as inputs current and proposed, and outputs the ratio of the posteriors of the proposed value to the current one (Hint: to do this in R you will need to use the following to calculate the value of the posterior at (x, y):

```
library (mvtnorm)
fGetRatio <- function(theta_current, theta_proposed) {
  r <- dmvnorm(theta_proposed, c(20, 5), sigma = Sigma) / dmvnorm(theta_current ,c(20, 5), sigma = Sigma

  if (ldebug == 1) cat("\nfGetRatio:r= ", r)

  return(r)

}
```

3

Create an accept/reject function which takes as inputs current and proposed, then uses the above ratio function to find: r = proposed ; then compares r with a uniformly-distributed current random number u between 0 and 1. If r > u = output proposed; otherwise output current

```
fAcceptReject <- function(theta_current, theta_proposed) {
  r <- fGetRatio(theta_current, theta_proposed)

  u <- runif(1,0,1)
  if ( r > u)
   return(theta_proposed)
  else
   return(theta_current)
}
```

4

Combine the proposal function along with the accept/reject function to make a function that takes as input current, proposes a new value of , then based on r moves to that new point or stays in the current position.

```
fSample <- function(theta_current) {
  if (ldebug == 1) cat("\nfSample: theta_current= ", theta_current)

  theta_proposed <- fproposal(theta_current)
  if (ldebug == 1) cat("\nfSample:theta_proposed= ", theta_proposed)

  theta_accepted <- fAcceptReject(theta_current, theta_proposed)
  if (ldebug == 1) cat("\nfSample:theta_accepted= ", theta_accepted)

  return(theta_accepted)

}
```

5

Create a function called "RWMetropolis" that takes a starting value of and runs for n steps

```
RWMetropolis <- function(theta, N) {
  if (ldebug == 1) cat("\nRWMetropolis: theta = ", theta)
  ltheta <- matrix(nrow = N, ncol = 2)
  ltheta[1,] <- theta

  for (t in 2:N){
    ltheta[t,] <- fSample(ltheta[t-1,])
  }

  return(ltheta)
}
```

```
theta_starting <- c(15,3)
lRWMsamples <- RWMetropolis(theta_starting, 100)
```

```
plot(lRWMsamples, type = 'l')
```