

Homework 5

DATA604 Simulation and Modeling

Daniel Dittenhafer

April 10, 2016

1

At the start of each week, the condition of a machine is determined by measuring the amount of electrical current it uses. According to its amperage reading, the machine is categorized as being in one of the following four states: low, medium, high and failed. A machine in the low state has a probability of 0.05, 0.03, and 0.02 of being in the medium, high, or failed state, respectively, at the start of the next week. A machine in the medium state has a probability of 0.09 and 0.06 of being in the high or failed state, respectively, at the start of the next week (it cannot, by itself, go to the low state). And, a machine in the high state has a probability of 0.1 of being in the failed state at the start of the next week (it cannot, by itself, go to the low or medium state). If a machine is in the failed state at the start of a week, repair is immediately begun on the machine so that it will (with probability 1) be in the low state at the start of the following week. Let X be a Markov chain where X_n is the state of the machine at the start of week n .

a) Given the Markov transition matrix for X .

The following code creates and prints the Markov transition matrix for X .

```
tmX <- matrix(c(0.90, 0.05, 0.03, 0.02,
                0, 0.85, 0.09, 0.06,
                0, 0.00, 0.90, 0.10,
                1, 0.00, 0.00, 0.00), nrow=4, byrow=TRUE)
kable(tmX)
```

0.9	0.05	0.03	0.02
0.0	0.85	0.09	0.06
0.0	0.00	0.90	0.10
1.0	0.00	0.00	0.00

b) A new machine always starts in the low state. What is the probability that the machine is in the failed state three weeks after it is new?

First we define a helper function:

```
markovChainStep <- function(initialState, tranMatrix, steps)
{
  n <- steps
  step <- initialState
  for(i in 1:n)
  {
    step <- step %*% tranMatrix
  }
  return (step)
}
```

Next we call the helper function to compute the requested probability. We step twice in the function as it is 2 steps beyond the initial probability state.

```
bStep <- markovChainStep(tmX, tmX, 2)
kable(bStep)
```

0.771	0.115875	0.085425	0.0277
0.114	0.617125	0.208575	0.0603
0.180	0.005000	0.732000	0.0830
0.830	0.087500	0.058500	0.0240

The probability that the machine is in the failed state 3 weeks after it is new is 0.0277.

c) What is the probability that a machine has at least one failure three weeks after it is new?

Compute the steady state probabilities for this recurrent transition matrix.

```
steadyState <- markovChainStep(tmX, tmX, 100)
steadyState
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.4918033 0.1639344 0.295082 0.04918033
## [2,] 0.4918033 0.1639344 0.295082 0.04918033
## [3,] 0.4918033 0.1639344 0.295082 0.04918033
## [4,] 0.4918033 0.1639344 0.295082 0.04918033
```

```
steadyState[1,4]
```

```
## [1] 0.04918033
```

```
Fm <- matrix(c(1, 1, 1, 1,
               1, 1, 1, 1,
               1, 1, 1, 1,
               1, 1, 1, 1), byrow=TRUE, nrow=4)
```

```
Fm
```

```
##           [,1] [,2] [,3] [,4]
## [1,]      1      1      1      1
## [2,]      1      1      1      1
## [3,]      1      1      1      1
## [4,]      1      1      1      1
```

```
Rm <- matrix(nrow=4, ncol=4)
for(i in 1:nrow(Fm))
{
  for(j in 1:ncol(Fm))
  {
    if(i == j)
    {
      Rm[i,j] <- 1 / (1 - Fm[j,j])
    }
    else
    {
      Rm[i,j] <- Fm[i,j] / (1 - Fm[j,j])
    }
  }
}
```

```
Rm
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  Inf  Inf  Inf  Inf
## [2,]  Inf  Inf  Inf  Inf
## [3,]  Inf  Inf  Inf  Inf
## [4,]  Inf  Inf  Inf  Inf
```

d) What is the expected number of weeks after a new machine is installed until the first failure occurs?

```
1 / steadyState[1,4]
```

```
## [1] 20.33333
```

e) On average, how many weeks per year is the machine working?

```
# Use Metropolis-Hastings to estimate?
```

f) Each week that the machine is in the low state...

a profit of \$1000 is realized; each week that the machine is in the medium state, a profit of \$500 is realized; each week that the machine is in the high state, a profit of \$400 is realized... What is the long-run average profit per week realized by the machine?

```
PL <- c(1000, 500, 400, -700)
PL
```

```
## [1] 1000 500 400 -700
```

```
lrAvgProfit <- PL %*% steadyState[1, ]
lrAvgProfit
```

```
##      [,1]
## [1,] 657.3771
```

g) Policy change?

Define the new profit and loss values:

```
newPL <- c(1000, 500, -600, -700)
newPL
```

```
## [1] 1000 500 -600 -700
```

Define the new transition matrix:

```
newX <- matrix(c(0.90, 0.05, 0.03, 0.02,
                 0, 0.85, 0.09, 0.06,
                 1, 0.00, 0.00, 0.00,
                 1, 0.00, 0.00, 0.00), nrow=4, byrow=TRUE)
kable(newX)
```

0.9	0.05	0.03	0.02
0.0	0.85	0.09	0.06
1.0	0.00	0.00	0.00
1.0	0.00	0.00	0.00

Compute the new steady state and long-run average profit:

```
newSteadyState <- markovChainStep(newX, newX, 100)

newLrAvgProfit <- newPL %*% newSteadyState[1, ]
newLrAvgProfit
```

```
##           [,1]
## [1,] 769.3023
```

At an average profit of \$ 769.3, an ≈ 111.93 increase, the new policy is worthwhile.

2

Rao (1973) presented an example on genetic linkage of 197 animals in four categories. The group sizes are (125, 18, 20, 34). Assume that the probabilities of the corresponding multinomial distribution are ...

```
groupSizes <- c(125, 18, 20, 34)

w <- 0.25
m <- 5000
burn <- 1000
#days <- 250
x <- numeric(m)

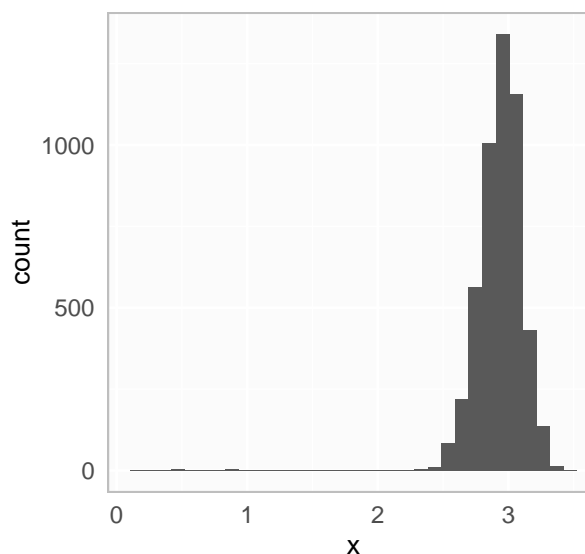
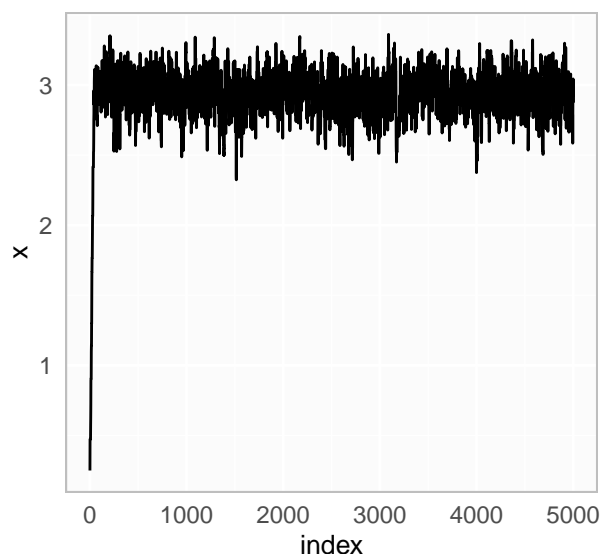
# Function to wrap the computation of the target density.
prob <- function(theta, K) {
  p <- (0.5 + (theta / 4))^K[1] * (1 - theta / 4)^K[2] * (1 - theta / 4)^K[3] * (theta/4)^K[4]
}

# Random Walk - Metropolis algorithm
u <- runif(m)
v <- runif(m, -w, w)
x[1] <- 0.25
for(i in 2:m) {
  y <- x[i-1] + v[i]
  if(u[i] <= prob(y, groupSizes) / prob(x[i-1], groupSizes)) {
    x[i] <- y
  } else {
    x[i] <- x[i-1]
  }
}

# Use mean to distill the MCMC results to a single value
xb <- x[burn + 1: m]
theta.hat <- mean(xb, na.rm = TRUE)
print(theta.hat)
```

```
## [1] 2.93708
```

The estimate of θ is 2.9370804. The following charts show the random walk Metropolis chain and resulting empirical distribution.



3

Define the provided Y vector:

```
Y <- c(4,5,4,1,0,4,3,4,0,6,3,3,4,0,2,6,3,3,5,4,5,3,1,4,4,1,5,5,3,4,2,5,2,2,3,4,2,1,3,2,2,1,1,1,1,3,0,0,1,0,1,
```

Define the fullcondm function per the implementation pseudo code.

```
fullcondm <- function(m, lambda, phi, y, n, alpha0, beta0, gamma0, delta0)
{
  lamexp <- ifelse(m > 1, sum(y[1:m]), 0)
  phiexp <- ifelse(m < n, sum(y[m+1:n]), 0)

  return (lambda^(alpha0-1+lamexp)*exp(-(beta0+m)*lambda)*phi^(gamma0-1+phiexp)*exp(-(delta0+n-m)*phi))
}
```

Define the R code for the Gibbs sample based on the pseudo-code for implementation:

```
alpha0 <- 1
gamma0 <- 1

n <- length(Y)
z <- 1000
lambda <- phi <- m <- B <- delta <- numeric(z)
#L <- numeric(n)

lambda[1] <- 1
phi[1] <- 1
m <- sample(1:n, 1)
B[1] <- 1
delta[1] <- 1

Mprob <- numeric(n)
mChain <- matrix(nrow=z, ncol=5)

# Gibbs Sampler
for(i in 1:z) {
```

```

sum1 <- sum(Y[1:m])
sum2 <- sum(Y[(m+1):n])

lambda[i] <- rgamma(1, sum1+alpha0, B[i] / (B[i] * m + 1))
phi[i] <- rgamma(1, sum2+gamma0, delta[i] / (delta[i] * (n-m) + 1))

for(j in 1:n) {
  Mprob[j] <- fullcondm(j, lambda[i], phi[i], Y, n, alpha0, B[i], gamma0, delta[i])
}

m <- sample(1:n, 1)

B[i+1] <- 1 / rgamma(1, alpha0, lambda[i] + 1)
delta[i+1] <- 1 / rgamma(1, gamma0, phi[i] + 1)

mChain[i,] <- c(lambda[i], phi[i], m, B[i], delta[i])
}

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```

```

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

```

```

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

```



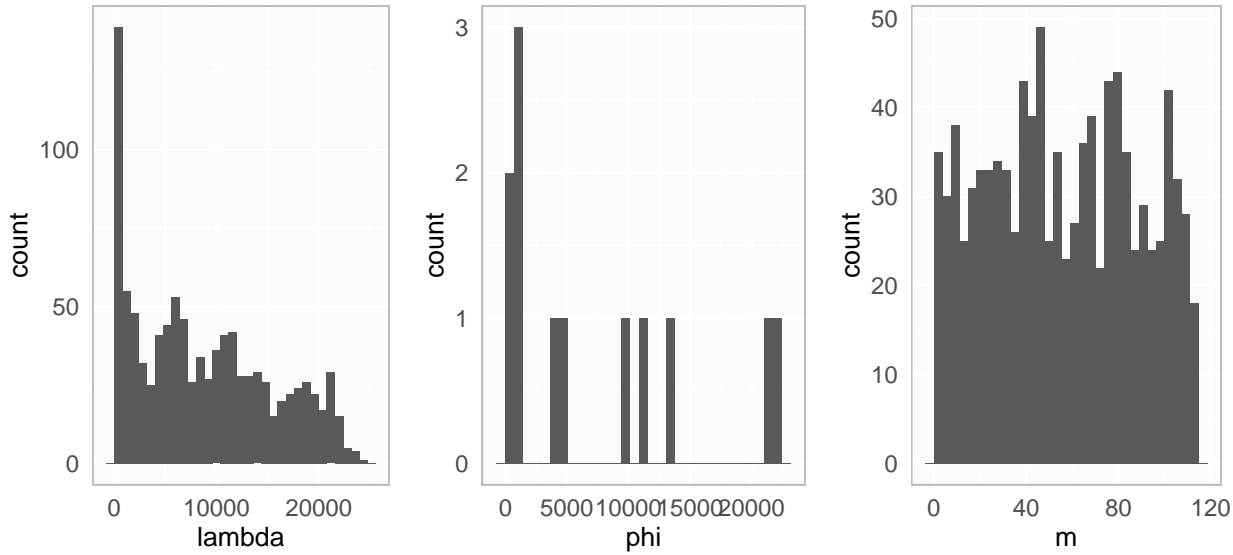
```
## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

## Warning in rgamma(1, sum2 + gamma0, delta[i]/(delta[i] * (n - m) + 1)): NAs
## produced

## Warning in rgamma(1, gamma0, phi[i] + 1): NAs produced

## Warning: Removed 988 rows containing non-finite values (stat_bin).
```



Experiment with Citations

The citation: (Mau, Newton, and Larget, 1999)

(Mau et al., 1999)

References

Mau, B., Newton, M.A. and Larget, B. (1999), “Bayesian phylogenetic inference via markov chain monte carlo methods”, *Biometrics*, Wiley-Blackwell, Vol. 55 No. 1, pp. 1–12.