# Second Assignment - MCMC, Poisson processes, Continuous time Markov chains

Cesar Conejo Villalobos, Josu Esarte Araiz, Cecilia Gallego Carracedo, Xavier Bryant (Group 4)

11/01/2021

## Question 1

[2.5 points] Go through Example 5.3 in Dobrow to learn how to use a Metropolis-Hastings algorithm to decode a message. Follow the described procedure to decode the message in row `i` of `messages.txt`, where `i` is your group number. The file `Englishcharacters.txt` contains the absolute frequencies of pairs of consecutive characters in English language (taken from the complete works of Jane Austen). Return the decoded message and provide the code with your implementation of the algorithm.

**Solution:** In this case, we are the group $i = 4$ and the encrypted message is given by table 1. This character will be saved under the variable name `codemess`.

Table 1: Encrypted message line 4

| x |
|---|
| nzfartb njl vlmlipgultn pc uptnl afzip ulnjpvh kl krii fihp szrlcio ultnrpt kjfn kl urbjn afii njl hlaptv bltlzfnrpt uaua zlmpiwnrpt hnfznrtb rt njl urvfnpfifnl trtlnrlh njrh rtaiwvlh njl vlmlipgultn pc gfznrail crinlzh zlmlzhrsil qwug ftv glzclan hfugirtb ftv aptaiwvlh krnj upzl awzzltn kpzx pt gpgwifnrpt pz hldwltnrfi uptnl afzip ftv zlbltlzfnrpt ftv njl apugwnrtb pc jptlhn hnftvfzv lzzpzh |

Using as reference the Example 5.3 of Dobrow and the script file `decode.R` available in the page author people.carleton.edu we will decrypt the message.

First, we denote the transition matrix $M$ as the absolute frequencies of pairs of consecutive characters contained in the file `Englishcharacters.txt`.

Then, we must consider that our encoded message has 394 characters. We will denoted them as $(c_1, ...c_{394})$. So, for each coding function, there is an associated score taking as reference the product over all the pairs of letters in the message with the number of recurrence of that pair in the matrix $M$. This score is given by the following formula:

$$\text{score}(f) = \prod_{i=1}^{393} M_{f(c_i),f(c_{i+1})}$$

This `score` function is provided by the author:

```r
# Computes the score of the decoded message using the given code.
# logmat is the logarithm of the values of transition matrix M (EnglishCharacters.txt)

score <- function(code)
{
  p <- 0
  # For each pair of letters in the decoded message
  # query the transition matrix for the probability of that pair
  for (i in 1:(nchar(message)-1)){
    p <- p + logmat[charIndex(substr(code, i, i)),charIndex(substr(code, i+1, i+1))]
  }
  # return the sum of these probabilities
  p
}
```

Finally, the algorithm for decryption of the message has the following steps:

1. Start with any coding function $f$. For convenience, use the identity.
2. Pick two letters uniformly at random and switch the values that $f$ assigns to these two letters. Call the new proposal function $f^*$.
3. Compute the acceptance function $a(f, f^*) = \frac{score(f^*)}{score(f)}$
4. Let $U$ be uniformly distributes on $(0, 1)$. If $U \leq a(f, f^*)$ accept $f^*$. Otherwise, stay with $f$.

Following step 1, the coded message provided in table 1 has a score of 2458.89. The implementation of the Metropolis-Hastings algorithm is given in the following code. Fixing the seed as the number of the iteration guarantees that 156000 iterations are enough for decrypting the message.

```
# Metropolis-Hastings algorithm

# Step 1: Start with any coding function. Identity for convenience.
curFunc <- 1:27
score_f <- score(decrypt(codemess,curFunc))

# instantiate a map to hold previously computed codes scores
map <- new.env(hash=T, parent=emptyenv())
map[[paste(curFunc, collapse='')]] <- score_f



# run 156000 iterations of the Metropolis-Hastings algorithm
for (iteration in 1:156000) {

  # Step 2: sample two letters to swap uniformly
  set.seed(iteration)
  swaps <- sample(1:26,2)

  # let curFunc be oldFunc but with two letters swapped
  oldFunc <- curFunc
  curFunc[swaps[1]] <- oldFunc[swaps[2]]
  curFunc[swaps[2]] <- oldFunc[swaps[1]]

  # Step 3. Compute the acceptance function: Two scenarios:
  # 3.1 if we have already scored this decoding,
  # retrieve score from our map
  if (exists(paste(curFunc, collapse =''), map)){
    score_f_star <- map[[paste(curFunc, collapse ='')]]
  } else
    # 3.2 if we have not already scored this decoding,
    # calculate it and store it in the map
  {
    score_f_star <- score (decrypt(codemess,curFunc))
    map[[paste(curFunc, collapse = '')]] <- score_f_star
  }
  # acceptance function
  acc <- exp(score_f_star-score_f)

  # Step 4. Accept the proposal function f* or keep f.
  if (runif(1) < acc)
  {
    score_f <- score_f_star

  } else
  {

    curFunc <- oldFunc

  }

  # print out our decryption every 2000 iterations
  if ((iteration %% 2000) == 0)
  {
    descrypted_text <- decrypt(codemess,curFunc)
    print(c(iteration,descrypted_text, score_f_star))
    descrypted_msg[[pos]] <- descrypted_text
    pos <- pos + 1
  }
}
```

The decrypted message is given by the table 2. It has a final score of 3868.9.

Table 2: Decrypted message

| x |
|---|
| tracing the development of monte carlo methods we will also briefly mention what we might call the second generation mcmc revolution starting in the midatoalate nineties this includes the development of particle filters reversible jump and perfect sampling and concludes with more current work on population or sequential monte carlo and regeneration and the computing of honest standard errors |

## Question 2

[3 points] The number of cars arriving to the university parking every morning follows a non-homogeneous Poisson process with a constant average of 100 cars per hour between 8 and 8:30 that increases linearly to 250 between 8:30 and 8:45 and continues to increase to 350 between 8:45 and 9 to then decrease to 100 per hour between 9 and 9:30. The parking has 150 spots and opens every morning at 8 a.m. with no car inside.

**a)** [0.75 points] Find the expectation of the number of cars that get to the university parking by time $t$.

**Solution:** Based on the description of the process, we have the following intensity function:

$$\lambda(t) = \begin{cases} 100 & \text{for } 0 \le t \le 0.5 \\ 600t - 200 & \text{for } 0.5 < t \le 0.75 \\ 400t - 50 & \text{for } 0.75 < t \le 1 \\ -500t + 850 & \text{for } 1 < t \le 1.5 \end{cases}$$

where $t$ represents the time fraction starting with $t = 0$ equivalent to $8:00$ am. Figure 1 provide a graphical representation.

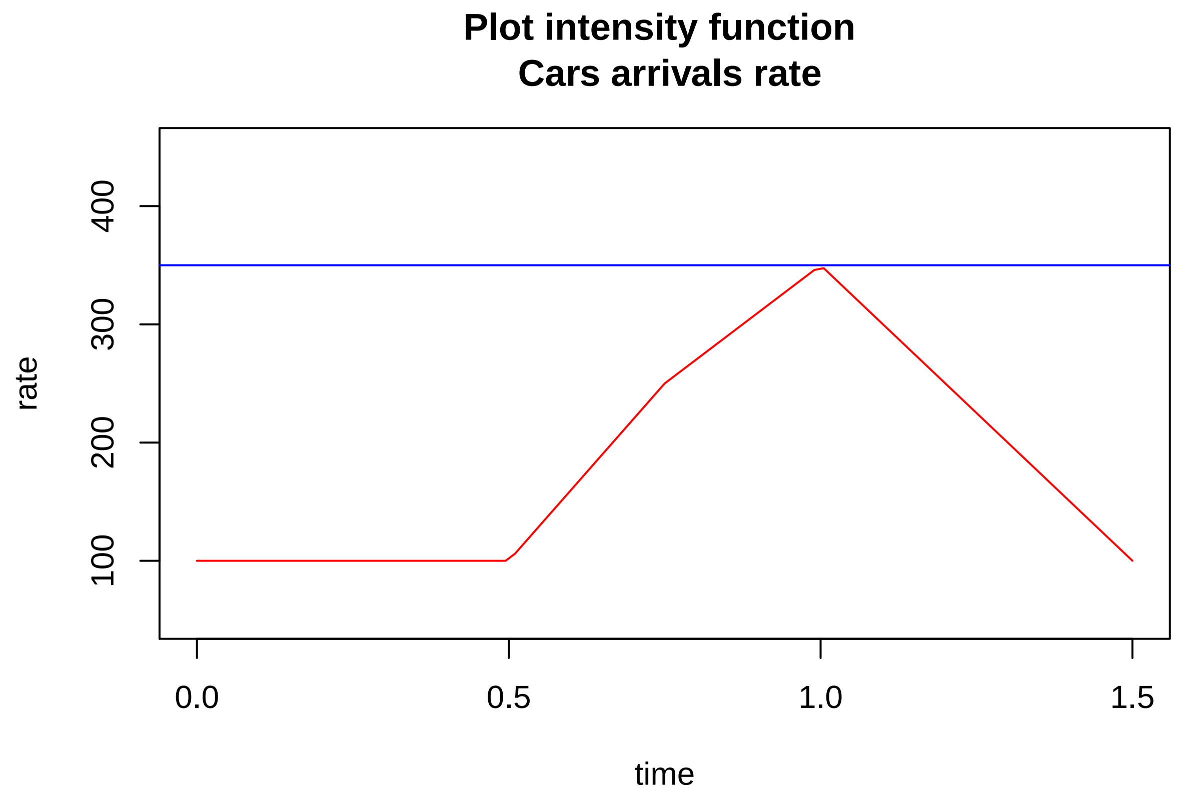

Figure 1: Intensity rate Cars arrival: Time starts at 8:00 am.

Define $N_t$ as the number of cars that get to the parking by time $t$. The expected number of cars that get to the parking by time $t$ is given $E[N_t] = \int_0^{1.5} \lambda(t)dt$ where:

$$\int_0^{1.5} \lambda(t)dt = \begin{cases} 50 & \text{for } 0 \le t \le 0.5 \\ 43.75 & \text{for } 0.5 < t \le 0.75 \\ 75 & \text{for } 0.75 < t \le 1 \\ 112.5 & \text{for } 1 < t \le 1.5 \end{cases}$$

So, the total expected number of car from 8 to 9:30 am is 281.25.

**b)** [0.5 points] Assuming that no car leaves before 9:30, use `R` (without simulation) to find the time you should get to the parking in order to find a free spot at least 80% of the days.

**c)** [1 point] Write an `R` function to simulate this non-homogenueous Poisson process from an homogeneous Poisson process.

**d)** [0.75 points] Use simulation (with the previously created function) to check your answer for part b).

## Question 3

[4.5 points] Consider a supermarket with 2 cashiers. Customers arrive to the unique cashiers waiting line according to a Poisson process with rate $\lambda$. The times to be served (check-out times at the cashier) are independent and distributed as exponential random variables with rate $\mu$. Let us assume that the queue can accomodate an unlimited number of waiting customers. This system is known as the M/M/2 queue. Let $X_t$ denote the number of customers in the system (at the cashiers or in the waiting line) at time $t$.

**a)** [0.75 points] What kind of process is $X = \{X_t, t \geq 0\}$? Write its state space and infinitesimal generator.

Based on the description in the previous statement, we have the following facts:

- Arrivals are modeled according to a Poisson Process, with rate $\lambda$. In this case, we know that the interarrival times are independent and exponentially distributed.
- The times to be served by the cashiers are independent and exponentially distributed with rate $\mu$.

From the previous statements, the minimun of this two events is also exponentially distributed, with parameter $\lambda + \mu$. As a result, this process can be modeled as a Continuous Time Markov Chain, where $X_t$ denote the number of customers in the system (at the cashiers or in the waiting line) at time $t$.

The space state for this process is given by $S = \{0\} \cup \mathbb{N}$. The infinitesimal generator $Q$ corresponds to:

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ \mu & -(\mu+\lambda) & \lambda & 0 & 0 & \cdots \\ 0 & 2\mu & -(2\mu+\lambda) & \lambda & 0 & \cdots \\ 0 & 0 & 2\mu & -(2\mu+\lambda) & \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

**b)** [1 point] Compute its stationary distribution. What is the expected number of people in the system (checking-out or in the waiting line) in the long-run?

The M/M/2 queue is a birth and death process with parameters:

- Arrival rate:

$$\lambda_i = \lambda$$

For all $i \geq 0$.

- Serving rate:

$$\mu_i = \begin{cases} i\mu & \text{for } i = 1,2 \\ 2\mu & \text{for } i \geq 3 \end{cases}$$

Then, we have that:

$$\begin{aligned} \sum_{k=0}^{\infty} \prod_{i=1}^{k} \frac{\lambda_{i-1}}{\mu_i} &= \sum_{k=0}^{1} \prod_{i=1}^{k} \frac{\lambda}{i\mu} + \sum_{k=2}^{\infty} \left( \prod_{i=1}^{k} \frac{\lambda}{i\mu} \right) \left( \prod_{i=1}^{k} \frac{\lambda}{2\mu} \right) \\ &= \sum_{k=0}^{1} \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!} + \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2 \sum_{k=2}^{\infty} \left(\frac{\lambda}{2\mu}\right)^{k-2} \\ &= 1 + \frac{\lambda}{\mu} + \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2 \sum_{j=0}^{\infty} \left(\frac{\lambda}{2\mu}\right)^{j} \end{aligned}$$

The infinity serie converge to $\frac{1}{1-\frac{\lambda}{2\mu}}$ if $0 < \lambda < 2\mu$. In this case, we have:

$$\sum_{k=0}^{\infty}\prod_{i=1}^{k}\frac{\lambda_{i-1}}{\mu_i} = 1+\frac{\lambda}{\mu}+\frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2\left(\frac{1}{1-\frac{\lambda}{2\mu}}\right)$$
$$= 1+\frac{\lambda}{\mu}+\frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2\left(\frac{2\mu}{2\mu-\lambda}\right)$$

Adding the condition $\sum_{k=0}^{\infty}\pi_k = 1$, the stationary distribution $\boldsymbol{\pi}$ exists with:

$$\pi_0 = \left[1+\frac{\lambda}{\mu}+\frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2\left(\frac{2\mu}{2\mu-\lambda}\right)\right]^{-1}$$

Finally, the stationary probabilities are given by $\pi_k = \pi_0\prod_{i=1}^{k}\frac{\lambda_{i-1}}{\mu_i}$ in the following way:

$$\pi_k = \begin{cases} \pi_0\left(\frac{\lambda}{\mu}\right)^k\frac{1}{k!} & \text{for } 0\le k<2 \\ \pi_0\left(\frac{\lambda}{\mu}\right)^k\frac{1}{2^{k-1}} & \text{for } k\ge 2 \end{cases}$$

With this expression, we can calculate the expected number of people in the system (checking-out or in the waiting line) in the long-run.

- Expected number of customers in the queue

**c)** [0.75 points] We say that *overtaking* occurs when a customer departs the supermarket before another customer who arrived earlier to the waiting line. In steady state, find the probability that an arriving customer overtakes another customer (you may assume that the state of the system at each arrival instant is distributed according to the stationary distribution).

**d)** [1 point] Write the `R` code necessary to simulate the system (provide the code) and generate the times customers leave the supermarket.

**e)** [1 point] Assuming 2 customers arrive to the cashiers every 5 minutes on average and it takes an average time of 4 minutes to check-out, estimate through simulation the probability of overtaking and compare it with the result you got in part c). Also estimate the long-run average number of people in the system and compare it with the result of part b). Explain all the considerations you make and the simulation setting (provide the code).

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that ge2