Stochastic Process of Gene Expression

Maryam Labaf

Gene Expression

Gene Expression is the process by which the information stored in DNA is converted to a functional product (e.g., Protein). Genes are transcriptionally regulated (e.g. turned on and off) to control the proteins production. Gene expression contains two main processes: *transcription* and *translation*. During transcription, the gene is copied to produce an RNA transcript called *messenger RNA(mRNA)*. In this process mRNA is synthesized with the genetic information transcribed from the gene. After the mRNA has carried the transcribed "message" from DNA, ribosomes -protein-making factories in the cell- bind to mRNA, decode the stored information and produce a specific amino acid chain which folds into an active protein.

Stochastic Model in Gene Expression

There are two mathematical models to explain the time evolution of a homogeneous chemical system: Deterministic and stochastic model. In deterministic model, the concentration of each chemical species is continuous and deterministic over time, and is represented by a set of ordinary differential equations. However, the stochastic process assumes the time evolution of the system is discrete and stochastic, such that two independent simulations of the same system would generate different time evolution trajectories.

The chemical reactions in the cell are all stochastic. We can not predict the cell's reaction process, when or where a reaction happens, deterministically. Although the deterministic continuous models work well when concentrations are large, in gene expression the number of molecules are typically small, i.e., the mRNA and protein levels are low because of low transcription and translation rate. Therefore, the deterministic approach may not be appropriate.So, we have to use the stochastic model for gene expression.

Markov Process

To compute the trajectory of the each state of a gene expression we need stochastic process. A Stochastic process is a collection of random variables which are associated by a set (e.g., time), such that each random variable of the stochastic process is uniquely indexed with an element in the set. A stochastic process is a *Markov Process* if the future behavior of the process depends only on its current position, not on how it got there. So, the chemical system (e.g., gene expression) has a markovian property as the reaction rates depends only on the current state of the system and it can be modeled as a continuous-time Markov chain known as *birth-death process*.

Master Equation

As discussed, all the fundamental gene expression processes (e.g. transcription, translation and decay of molecules) are considered as *Poisson processes*.(note: on the real line, the Poisson process is a type of birth-death process). So, the probability of occurrence of one reaction with rate k over Δt time interval is equal to $k\Delta t$.

Let $P(m,t)$ be the probability of having m number of mRNA at time t. Then, the evolution of mRNA can be written in the following:

$$P(m,t+\Delta t) = k_m P(m-1,t)\Delta t + \mu_m(m+1)P(m+1,t)\Delta t \quad (1)$$
$$+ [1 - k_m\Delta t - \mu_m m\Delta t]P(m,t) \quad (2)$$

Where k_m is the rate of mRNA production and μ_m is the rate of mRNA degradation. The first term of the equation 1 is the probability of mRNA production, second term is the probability of mRNA degradation, and the last term is the probability that the system stays in the m mRNA state.

So, the rate of number of the mRNA remaining in the system between time t and $t + \Delta t$:

$$\frac{P(m, t + \Delta t) - P(m, t)}{\Delta t} = k_m P(m-1, t) + \mu_m (m+1) P(m+1, t) - (k_m + \mu_m m) P(m, t)$$

In the limit that $\Delta t \sim \partial t \to 0$, the partial differential equation is equal to:

$$\frac{\partial P(m, t)}{\partial t} = \underbrace{k_m P(m-1, t) + \mu_m (m+1) P(m+1, t)}_{\text{gain term}} \tag{3}$$

$$- \underbrace{(k_m + \mu_m m) P(m, t)}_{\text{loss term}} \tag{4}$$

The goal is solving equation 3 for $P(m, t)$ in order to compute the number of mRNA in the system at time t. This can be achieved using generating functions. Define the generating function $G(z, t) = \sum_{m=0}^{\infty} z^m P(m, t)$. Differentiating with respect to t, we obtain

$$\frac{\partial}{\partial t} \sum_{m=0}^{\infty} z^m P(m, t) = \sum_{m=0}^{\infty} z^m \big[k_m P(m-1, t) + \mu_m (m+1) P(m+1, t) \tag{5}$$

$$- (k_m + \mu_m m) P(m, t) \big] \tag{6}$$

Computing each part of the equation:

1. $< LHS >$

$$\frac{\partial}{\partial t} \sum_{m=0}^{\infty} z^m P(m, t) = \frac{\partial}{\partial t} G(z, t)$$

2. $< RHS >$

$$\sum_{m=0}^{\infty} z^m k_m P(m-1, t) = k_m \sum_{m=0}^{\infty} z^{m+1} P(m, t) = z k_m G(z, t)$$

$$\sum_{m=0}^{\infty} z^m \mu_m (m+1) P(m+1, t) = \mu_m \sum_{m=0}^{\infty} m z^{m-1} P(m, t) = \mu_m \frac{\partial}{\partial z} G(z, t)$$

$$\sum_{m=0}^{\infty} z^m k_m P(m, t) = k_m G(z, t)$$

$$\sum_{m=0}^{\infty} z^m \mu_m m P(m, t) = z \mu_m \sum_{m=0}^{\infty} m z^{m-1} P(m, t) = z \mu_m \frac{\partial}{\partial z} G(z, t)$$

So, we could rewrite equation 5 as:

$$\frac{\partial}{\partial t} G(z, t) = k_m (z-1) G(z, t) + \mu_m (1 - z) \frac{\partial}{\partial z} G(z, t) \tag{7}$$

Solving equation 7 at steady state condition, when $\frac{\partial}{\partial t} G \to 0$:

$$(z-1) \big[k_m G(z, t) - \mu_m \frac{\partial}{\partial z} G(z, t) \big] = 0$$

If $z = 1$ then $G(1, t) = 1$. In order to find $P(m, t)$ we need to solve $k_m G(z, t) - \mu_m \frac{\partial}{\partial z} G(z, t) = 0$ for $G(z, t)$.

$$G(z, t) = e^{\frac{k_m}{\mu_m}(z-1)} \tag{8}$$

Using the $G(z, t) = \sum_{m=0}^{\infty} z^m P(m, t)$, and the Taylor series of e^x, the steady state solution to equation 3 is a Poisson distribution as:

$$P(m, t) = \frac{1}{m!} (\frac{k_m}{\mu_m})^m e^{-\frac{k_m}{\mu_m}} \tag{9}$$

Mean value and Variance

At steady state condition the mean value is $< m >= \sum_{m=0}^{\infty} mP(m,t)$ and variance is $< \sigma^2 >=< m^2 > - < m >^2$. We could use the generating function G in order to find the mean and standard deviation at steady state condition in the following:

$$\begin{aligned} G(z,t) &= \sum_{m=0}^{\infty} z^m P(m,t) \\ \frac{\partial G}{\partial z} &= \sum_{m=1}^{\infty} m z^{m-1} P(m,t) \\ \frac{\partial G}{\partial z}\Big|_{z=1} &= \sum_{m=0}^{\infty} mP(m,t) \\ &=< m > \end{aligned}$$

Also:

$$\begin{aligned} G(z,t) &= e^{\frac{k_m}{\mu_m}(z-1)} \\ \frac{\partial G}{\partial z} &= \frac{k_m}{\mu_m} e^{\frac{k_m}{\mu_m}(z-1)} \\ \frac{\partial G}{\partial z}\Big|_{z=1} &= \frac{k_m}{\mu_m} \\ &=< m > \end{aligned}$$

Therefore, $< m >= \frac{k_m}{\mu_m}$. To compute the variance, we need to find the second derivative of $G(z,t)$(note: $< m^2 >= \sum_{m=0}^{\infty} m^2 P(m,t)$). Repeating the above computations we could find variance $< \sigma^2 >= \frac{k_m}{\mu_m}$. Standard deviation σ can be derived from variance as well.

Gillespie Algorithm for Modeling Stochastic Systems

Master equation in most of the cases is too complex to be solved. So, we could use simulation algorithms as a numerical scheme that is equivalent to solving the master equation. Gillespie algorithm is a good simulation method to test the analytical result of the derived equations of the stochastic gene expression. The essential assumption of this algorithm is that all processes are considered as Poisson processes with some rate k, where $t\Delta t$ is the probability of occurring production during the small time interval Δt. The central point of the algorithm is drawing 2 random numbers at each time interval Δt. One random number to determine WHEN the next reaction among all the reactions will take place, and the second random number to choose WHICH one of the reactions will occur.

Implementing of Gillespie 's Algorithm

At every time step, r a random number is generated using a uniformly distribution function between $[0,1]$. The process will happen when r is not greater than $k\Delta t$. At time t the number of molecules of each of the interacting species are known, denote by $a_j(t)$, and $a_0(t) = \sum a_j(t)$.To increase the accuracy of the algorithm, the time interval should be chose small enough. So, we consider the waiting time for the Poisson process. This waiting time is a generated as a random number to measure the time pass from the last occurrence of the reaction until the time that it happens a gain. The distribution of time is exponential distribution. The system is known at time t. The probability that the next reaction occurs between $t+\tau$ and $t+\tau+\delta\tau$ is equal to:

$$\begin{aligned} P(\tau)\delta\tau &= P(\text{no reaction between } t \text{ and } t+\tau) P(\text{reaction between } t+\tau \text{ and } t+\tau+\delta\tau) \\ &= e^{(-a_0(t)\tau)} a_0(t)\delta\tau \end{aligned}$$

In order to simulate the system, we should take the following steps:

1. Generate a random number r_1 from an exponential probability distribution $P(\tau) = a_0 e^{-a_0\tau}$ of when the next reaction takes place.
2. Draw another random number,r_2 from uniform distribution $[0,1]$ to chose which reaction occur. If $0 < r_2 < \frac{a_1}{a_0}$ reaction 1 is chosen, if $\frac{a_1}{a_0} < r_2 < \frac{a_1+a_2}{a_0}$ reaction 2 is chosen and so on.

3. occurrence of each reaction can change the number of molecules involved in that reaction by one.
4. These process reiterate to follow the evolution of the system.

R code implementation of Gillespie' s algorithm

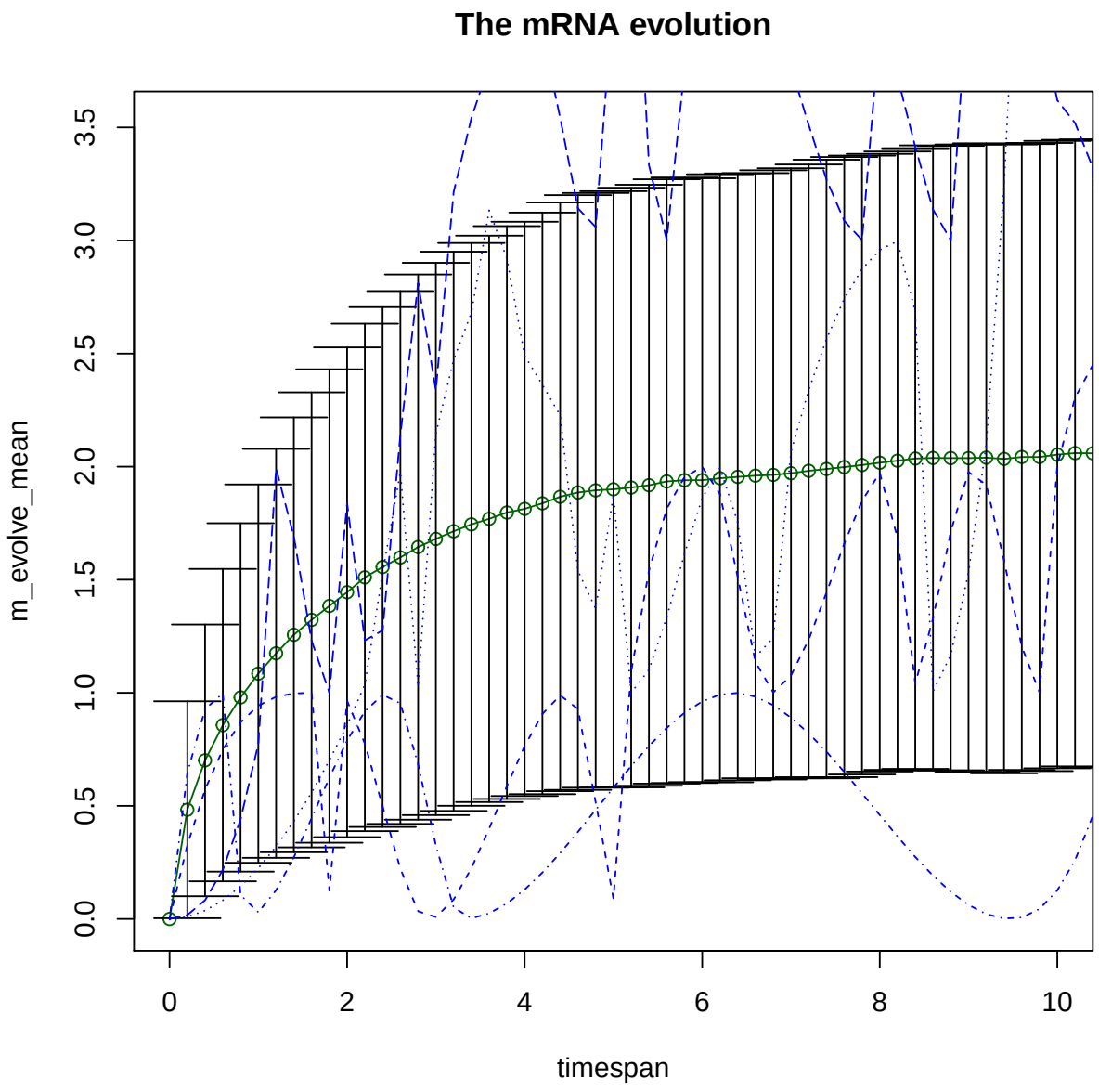
For the simple case of one DNA produce mRNA with rate k_m and mRNA decays with rate of μ_k, there are two reactions:

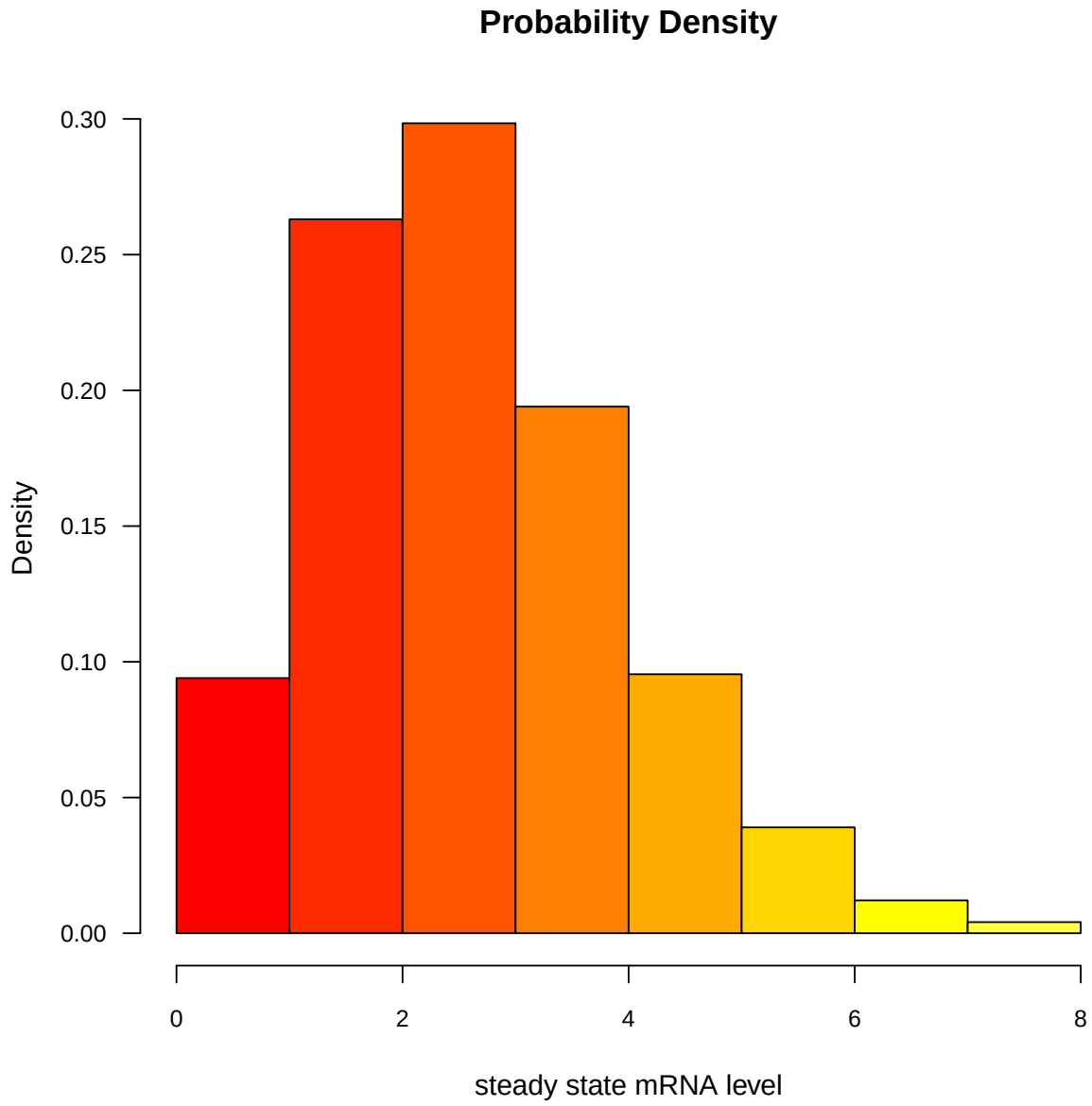
1. $D \rightarrow D + M$ with rate of k_m
2. $M \rightarrow$ with rate of μ_m

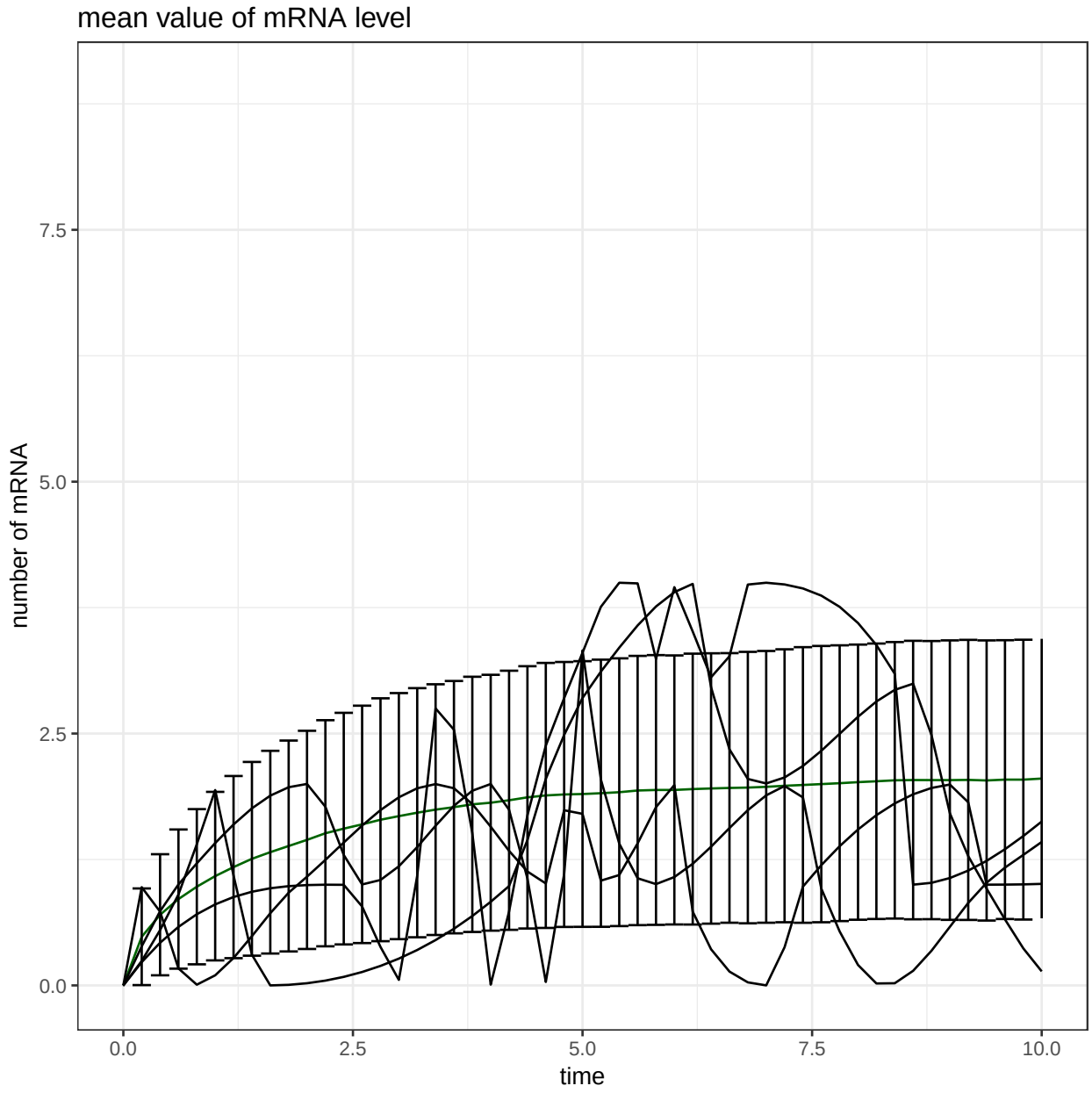
```
################################################
##Apply Gillespie Algorithm for gene regulation##
##for the simple case of one DNA produce mRNA####
################################################
#install.packages("pracma")
library(pracma)
library(ggplot2)
## intialization of components
N_cell <- 10000 #numebr of cells to simulate
timelimit <- 1000 #time of simulation for each cell
d <- 1 #numer of DNA molecules
k_m <- 1 #rate of mRNA production
mu_m <- 0.5
#rate of mRNA degeredation

#create a zero vector to save the number of last mRNA for each cell
m_end <- rep(0,N_cell)
timespan <- seq(0,timelimit,by = 0.2)
m_evolve = matrix(0,nrow =  N_cell, ncol = length(timespan))
dim(m_evolve)

## [1] 10000  5001

#loop over the numebr of cells
for (i in 1:N_cell){
  #print(c("Calculating species for cell ",i,N_cell))
  m <- 0
  time <-0
  m_tmp <-m
  t_tmp<- time

  while(time < timelimit){
    event_1 <- k_m*d
    event_2 <- mu_m*m

    e_0 <- event_1+event_2

    #r_1 WHEN transition happen
    r_1 <- runif(1)
    tau <- -(1/e_0)*log(r_1)
    #r_2 WHICH transition occures
    r_2 <- runif(1)
    y <- r_2*e_0

    ##cumulative probability for each transition
    cumprobs <- rep(0,3)
```

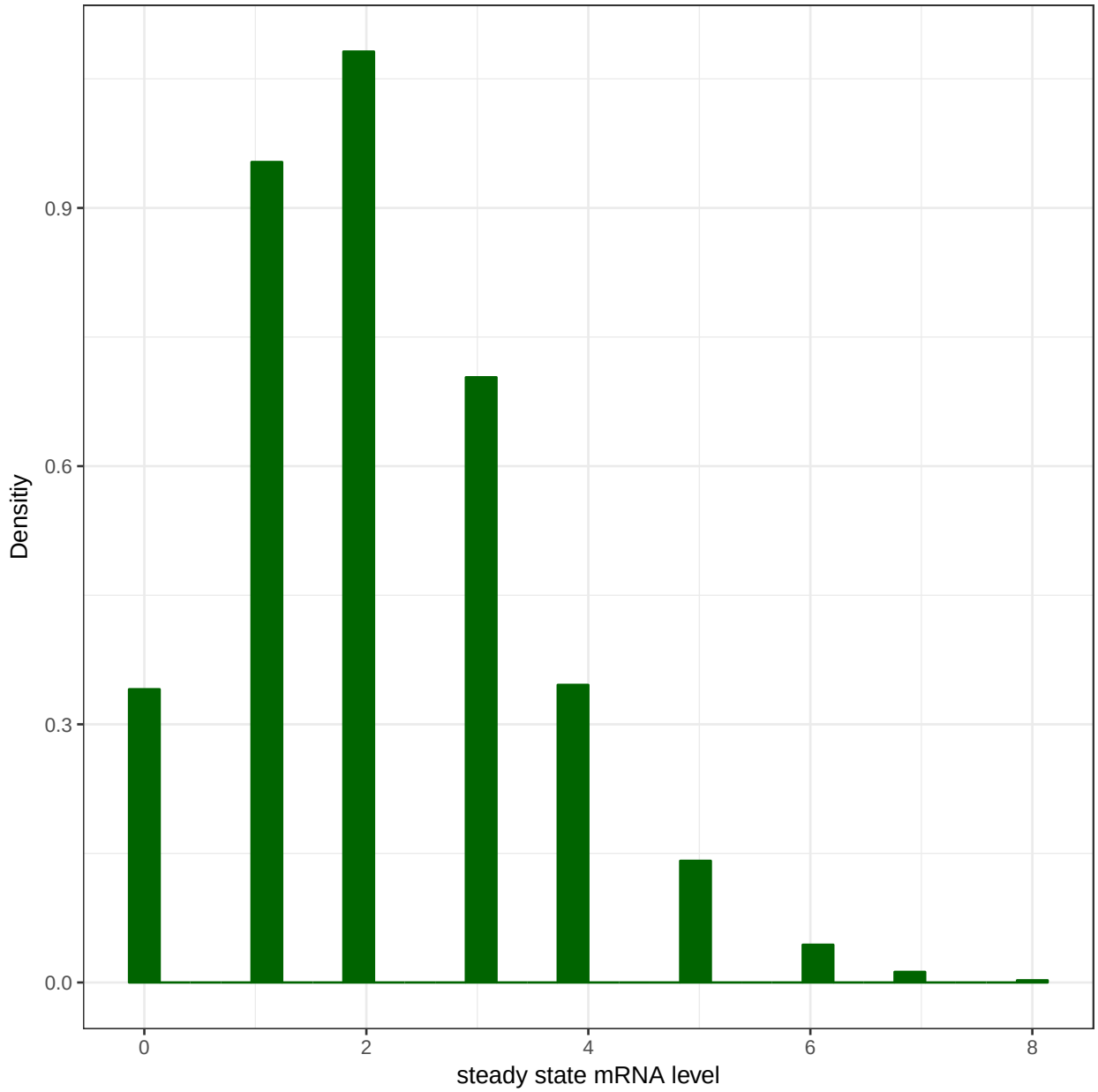
```
    cumprobs[2] <- cumprobs[1] +event_1
    cumprobs[3] <- cumprobs[2] +event_2
    for(j in 2:length(cumprobs)){
      if ((cumprobs[j] >= y) & (cumprobs[j-1] < y)){
        mu <- j-1
      }
    }
    if (mu == 1){
      m = m + 1
    }
    if (mu == 2){
      m = m - 1
    }
    time <- time+as.numeric(tau)
    #save time and numebr of mRNA at this stage
    m_tmp <- c(m_tmp,m)
    t_tmp <- c(t_tmp,time)
  }
  m_end[i] <- m
  #pchip: piecewise Cubic Hermite Interpolation
  m_evolve[i,] <- pchip(t_tmp, m_tmp, timespan)
}

m_evolve_mean <- apply(m_evolve, 2, mean)
m_evolve_sd <- apply(m_evolve,2,sd)
m_mean <- mean(m_end)
m_sd <- sd(m_end)
#############################################################################
##ploting with plot command
plot(timespan,m_evolve_mean,
     ylim = range(c(m_evolve_mean-m_evolve_sd, m_evolve_mean+m_evolve_sd)),
     type = "o", col = "darkgreen", xlim = c(0,10))
arrows(timespan,m_evolve_mean-m_evolve_sd, timespan, m_evolve_mean+m_evolve_sd,
       length = 0.2, angle = 90, code =3)
lines(timespan,m_evolve[5000,], col = "blue", lty = 2)
lines(timespan,m_evolve[5001,], col = "blue", lty = 3)
lines(timespan,m_evolve[5003,], col = "blue", lty = 4)
lines(timespan,m_evolve[5004,], col = "blue", lty = 5)
title("The mRNA evolution")

max_num <- max(m_end)
brk <- seq(0,max_num, by =1)
hist(m_end, col=heat.colors(length(brk)), breaks=brk,
     xlim=c(0,max_num), right=F,
     main="Probability Density",
     xlab = "steady state mRNA level",
     las=1, cex.axis=0.8, freq=F)
#############################################################################
#plotting with ggplot
m_evolve.df <- data.frame(t = timespan, evolveMean = m_evolve_mean)
gp <- ggplot(data = m_evolve.df, aes(x = t, y = evolveMean))+
      geom_line(color = 'darkgreen')+theme_bw()+ xlim(0,10)+
      geom_line(aes(x = timespan,y = m_evolve[3000,]),size = 0.5)+
      geom_line(aes(x = timespan,y = m_evolve[3001,]),size = 0.5)+
      geom_line(aes(x = timespan,y = m_evolve[3002,]),size = 0.5)+
      geom_line(aes(x = timespan,y = m_evolve[3003,]),size = 0.5)+
      geom_errorbar(mapping = aes(x = t,ymin = m_evolve_mean-m_evolve_sd,
                                  ymax = m_evolve_mean+m_evolve_sd),width = 0.2)+
```

```
      ggtitle("mean value of mRNA level") + labs(x="time", y="number of mRNA")
plot(gp)

m_end.df <- data.frame(Finalmean = m_end)
gp <- ggplot(data = m_end.df, aes(x = Finalmean, y = ..density..))+
  geom_histogram(position = "identity", color = 'darkgreen',fill = 'darkgreen')+
  theme_bw()+
  labs(x = "steady state mRNA level", y = "Densitiy")
plot(gp)
```




Probability Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0
2
4
6
8
steady state mRNA level


mean value of mRNA level
number of mRNA
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
time



The mean value and standard deviation are 2.1204 and 1.3620166 respectively, which is equivalent to what we extract from the formula.