Assignment 1, MVE550, autumn 2022

Lisa Samuelsson - *lisasam@student.chalmers.se*
Isak Palenius - *palenius@student.chalmers.se*
Oskar Giljegård - *oskargi@student.chalmers.se*

November 18, 2022

1. Assume that the number of requests for data that a computer server receives in an hour is Poisson(λ) distributed, where λ is the expected number of requests per hour. Assume that during each of 6 hours of operation, the number of requests were 2, 6, 3, 4, 3, and 3, respectively.

a) Using a prior that is proportional to $1/\lambda$, find the distribution for, and plot using R, the posterior distribution for λ given the data.

Let K be the list of events we observed, i.e. $K = 2, 6, 3, 4, 3, 3$. Furthermore, let A be a random variable corresponding to having seen all K events. We know that the number of request in an hour is Poisson distributed. We use a prior proportional to $\frac{1}{\lambda}$, $\pi(\lambda) \propto_\lambda 1/\lambda$. Using Bayes Theorem, the posterior can be written as

$$\pi(\lambda|A) = \frac{\pi(A|\lambda) \cdot \pi(\lambda)}{\pi(A)} \tag{1}$$

Since the number of requests are independent for a given lambda, we have that

$$\pi(A|\lambda) = \pi(2|\lambda) \cdot \pi(6|\lambda) \cdot ... \cdot \pi(3|\lambda) = \prod_{k \in K} \frac{e^{-\lambda}\lambda^k}{k!} \tag{2}$$

Using proportionality, we can finally write

$$\pi(\lambda|A) \propto_\lambda \pi(A|\lambda) \cdot \pi(\lambda) \propto_\lambda e^{-6\lambda} \cdot \lambda^{21} \cdot \frac{1}{\lambda} = e^{-6\lambda} \cdot \lambda^{20} \tag{3}$$

And so, we can see that the posterior is Gamma distributed,

$$\pi(\lambda|A) = Gamma(\lambda; 21, 6). \tag{4}$$

The following code was written to plot the posterior distribution, giving the plot shown in figure 1:

```
lambda_max = 8

x <- seq(0, lambda_max, by = 0.01)
plot(x, xlab="lambda",
ylab="probability", dgamma(x, shape = 21, rate = 6))
```

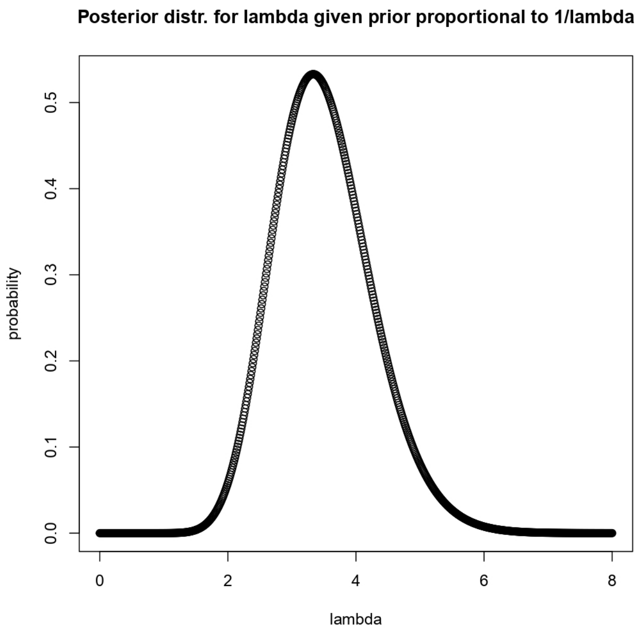

Figure 1: The posterior distribution for λ given a prior proportional to $\frac{1}{\lambda}$ given the data.

(b) Find the distribution for, and plot using R, the predictive distribution for the number of requests for the 7th hour, based on the given data.

Let k be the number of requests for the 7th hour. Let $\hat{\lambda} = \lambda | A$. From *a)*, we have

$$\pi(\hat{\lambda}) = Gamma(\alpha, \beta)$$

We also know that the likelihood is still Poisson distributed.

$$\pi(k|\hat{\lambda}) = Poisson(k; \hat{\lambda})$$

Then the posterior can be calculated as follows.

$$\begin{aligned}
\pi(\hat{\lambda}|k) &\propto_{\hat{\lambda}} \pi(k|\hat{\lambda})\pi(\hat{\lambda}) \\
&\propto_{\hat{\lambda}} e^{-\hat{\lambda}}\hat{\lambda}^k \; \hat{\lambda}^{\alpha-1}e^{-\beta\hat{\lambda}} \\
&= e^{-(\beta+1)\hat{\lambda}}\hat{\lambda}^{\alpha+k-1} \\
&\propto_{\hat{\lambda}} Gamma(\alpha+k, \beta+1)
\end{aligned}$$

The new likelihood then becomes

$$\pi(k|A) = \frac{\pi(k|\hat{\lambda}) \cdot \pi(\hat{\lambda})}{\pi(\hat{\lambda}|k)} = \frac{Poisson(k;\hat{\lambda})Gamma(\hat{\lambda};\alpha,\beta)}{Gamma(\hat{\lambda};\alpha+k,\beta+1)} = \tag{5}$$

$$\frac{e^{-\hat{\lambda}}\frac{\hat{\lambda}}{k!} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)}\hat{\lambda}^{\alpha-1}exp(-\beta\hat{\lambda})}{\frac{(\beta+k)^{\alpha+1}}{\Gamma(\alpha+k)}\hat{\lambda}^{\alpha+k-1}exp(-\beta\hat{\lambda}-\hat{\lambda})} = \frac{\beta^\alpha\Gamma(\alpha+k)}{(\beta+1)^{\alpha+k}\Gamma(\alpha)k!} \tag{6}$$

We get that the predicitve distribution $\pi(k)$ is a Negative Binomial(α, $\frac{\beta}{(1+\beta)}$) distribution.

We have $\alpha = 21$ and $\beta = 6$, so we get Negative Binomial(21, $\frac{6}{7}$)

The following code was written to plot the predictive distrubtion for the number of requests for the 7th hour, giving the plot shown in figure 2:

```
x <- seq(0, 16, by = 1)
plot(x, xlab="number of requests", ylab="probability",
dnbinom(x, size = 21, prob = 6/7))
```

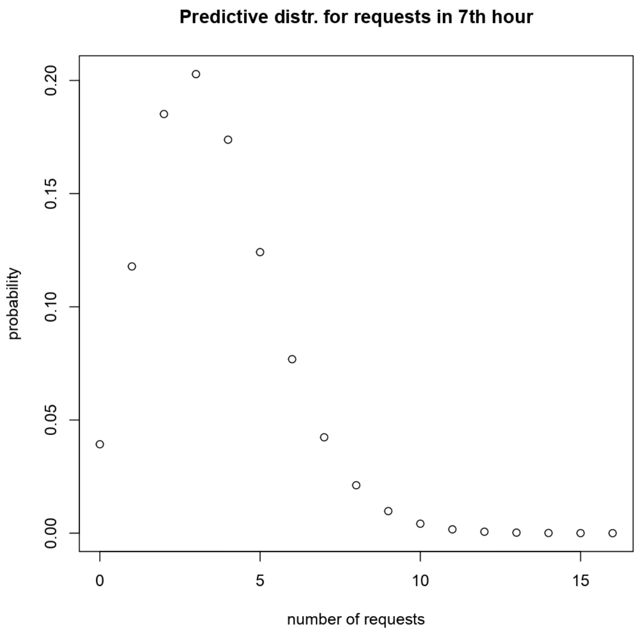

Figure 2: The plot shows the predictive distrubtion for the number of requests for the 7th hour.

(c) Compute and plot the posterior for λ using discretization.

Computing the posterior using discretization requires dividing up the continuous set of possible λ into discrete values. To do this we use a maximum and a minimum value for λ and then create k evenly spaced values in that range. A greater k leads to a higher resolution of the final graph. Ideally the max and min λ should be picked such that the prior is only positive within that interval. This is easy for something like a binomial distribution where p must be between 0 and 1. However, since our prior is proportional to $1/\lambda$ the minimum should be infitesimally close to 0 and the maximum should be $+\infty$. Therefore we chose 0.001 and 8 as min and max respectively which gives us a fairly good range from our prior. Since we know from part *a)* what the plot should look like we also know that the probability is very close to 0 outside the interval 2-6.

We then calculate the prior and the likelihood of all selected values for λ. Note that these are only proportionally correct and may be scaled incorrectly. For example, we are actually using $1/\lambda$ as the prior and not some function proportional to $1/\lambda$ as the question states. The posterior is proportional to the prior times the likelihood so it can be calculated by elementwise multiplication of our prior and likelihood vectors. Finally we scale the resulting posterior vector which fixes all the proportionality constants that we ignored.

The following code was written to plot the posterior distrubtion for λ of requests using discretization, giving the plot shown in figure 3:

```
lambda_max = 8

k <- 800
lambdas <- seq(0.001, lambda_max, length.out = k)
prior <- 1 / lambdas
placeholder_list <- seq(1, 1, length.out = k)
for (d in c(2, 6, 3, 4, 3, 3)) {
    likelihood_i <- dpois(d, lambdas)
    placeholder_list <- placeholder_list * likelihood_i
}

likelihood <- placeholder_list
posterior_distr <- prior * likelihood / sum(prior *
    likelihood) * k / lambda_max
plot(
    lambdas,
    posterior_distr,
    xlab="lambda",
    ylab="probability",
    main = "Posterior distr.
        for lambda using discretization"
)
```

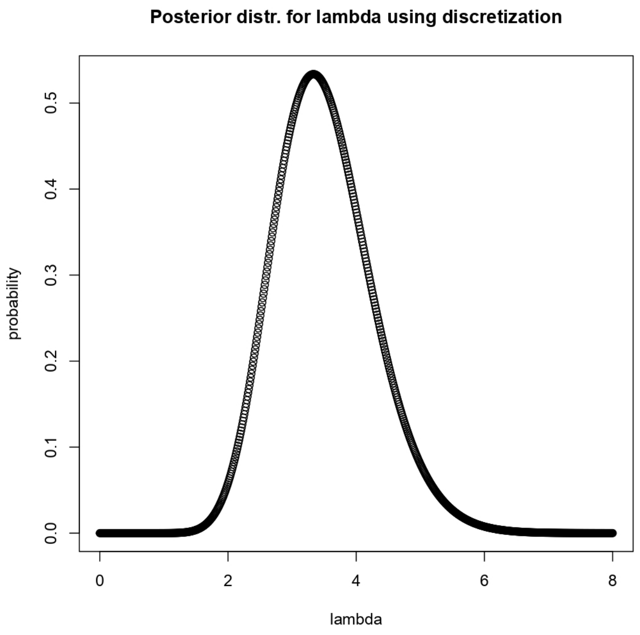

Figure 3: The posterior distribution for λ using discretization.

(d) By using information from similar servers, one realizes that prior information about λ can be expressed as a normal distribution with expectation 2.3 and standard deviation 1.7, restricted (truncated) to positive values. Using this new prior and discretization, plot using R the posterior for λ.

This task is exactly the same as task *c)* just with a different prior. Therefore we change from our previous prior, which was $1/\lambda$, to instead read off the density function of the normal distribution with mean 2.3 and standard deviation 1.7. Note that we could change the minimum lambda to 0 now since the normal distribution is not undefined at 0, however we chose to keep it the same as in *c)* for simplicity.

The following code was written to plot the posterior distrubtion for λ of requests using discretization using a normal distribution as prior, giving the plot shown in figure 4:

```
lambda_max = 8

k <- 800
lambdas <- seq(0.001, lambda_max, length.out = k)
prior <- dnorm(lambdas, mean = 2.3, sd = 1.7)
```

```
placeholder_list <- seq(1, 1, length.out = k)
for (d in c(2, 6, 3, 4, 3, 3)) {
    likelihood_i <- dpois(d, lambdas)
    placeholder_list <- placeholder_list * likelihood_i
}
likelihood <- placeholder_list
posterior_distr <- prior * likelihood / sum(prior
    * likelihood) * k / lambda_max
plot(
    lambdas,
    posterior_distr,
    xlab="lambda",
    ylab="probability",
    main = "Posterior distr.
        for lambda using normally distr. prior"
)
```

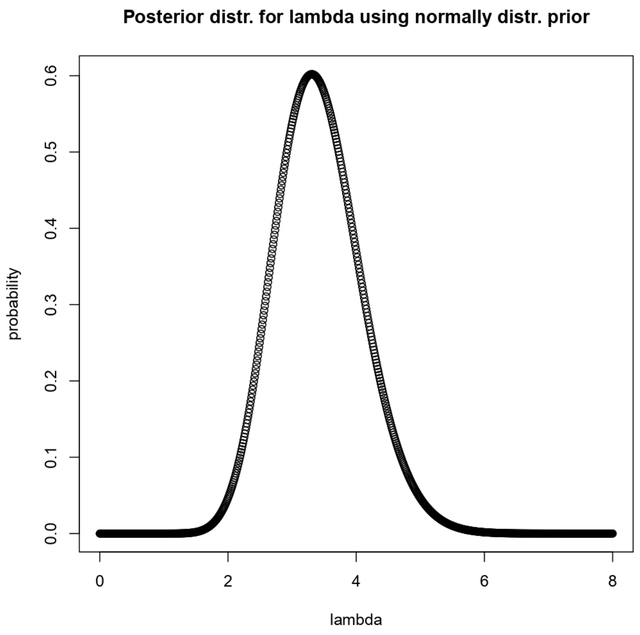

Figure 4: The posterior distribution for λ using discretization and a normally distributed prior.

(e) Using the new prior, compute using numerical integration in R the predictive probability for the server receiving exactly 3 requests during the 7th hour.

Using:

$$\begin{aligned}
\pi(A_{new}|A) &= \int_\lambda \pi(A_{new}|\lambda) \cdot \pi(\lambda|A)\, d\lambda \\
&= \int_\lambda \pi(A_{new}|\lambda) \cdot \frac{\pi(A|\lambda) \cdot \pi(\lambda)}{\int_\lambda \pi(A|\lambda) \cdot \pi(\lambda)\, d\lambda}\, d\lambda \\
&= \frac{\int_\lambda \pi(A_{new}|\lambda) \cdot \pi(A|\lambda) \cdot \pi(\lambda)}{\int_\lambda \pi(A|\lambda) \cdot \pi(\lambda)\, d\lambda}\, d\lambda
\end{aligned}$$

with:

$$\pi(A_{new}|\lambda) = \pi(3|\lambda) \tag{7}$$

$$\pi(A|\lambda) = \pi(2|\lambda) \cdot \pi(6|\lambda) \cdot ... \cdot \pi(3|\lambda) \tag{8}$$

$$\pi(\lambda) = \mathcal{N}(\mu = 2.3,\, \sigma = 1.7) \tag{9}$$

yields the result 0.2069919. In other words the probability for the server receiving exactly 3 requests during the 7th hour is approximately 20.7 %.

The following code was written to calculate the result:

```
lambda_max = 8

f1 <- function(lambda) {
  dpois(3, lambda) * dpois(2, lambda) * dpois(6, lambda)
  * dpois(3, lambda) * dpois(4, lambda) *
  dpois(3, lambda) * dpois(3, lambda) *
  dnorm(lambda, mean = 2.3, sd = 1.7)
}
f2 <- function(lambda) {
  dpois(2, lambda) * dpois(6, lambda) * dpois(3, lambda)
  * dpois(4, lambda) * dpois(3, lambda) *
  dpois(3, lambda) * dnorm(lambda, mean = 2.3, sd = 1.7)
}

numerator <- integrate(Vectorize(f1),
                       0, lambda_max)$value
denominator <- integrate(Vectorize(f2),
                         0, lambda_max)$value
result <- numerator / denominator
print("Predictive probability for receiving exactly 3
reqesust the 7th hour:")
print(result)
```

(f) Using the discretization for the posterior found in *(d)*, simulate and make a histogram for a sample of size 10 000 from the predictive distribution for the number of requests for the 7th hour.

From the calculated posterior distribution of λ, we used the function `sample()` to get a sequence of 10 000 λ's. For each λ, a Poisson distribution was calculated and a number of request was sampled from that distribution. The 10 000 samples resulted in the histogram shown in figure 5.

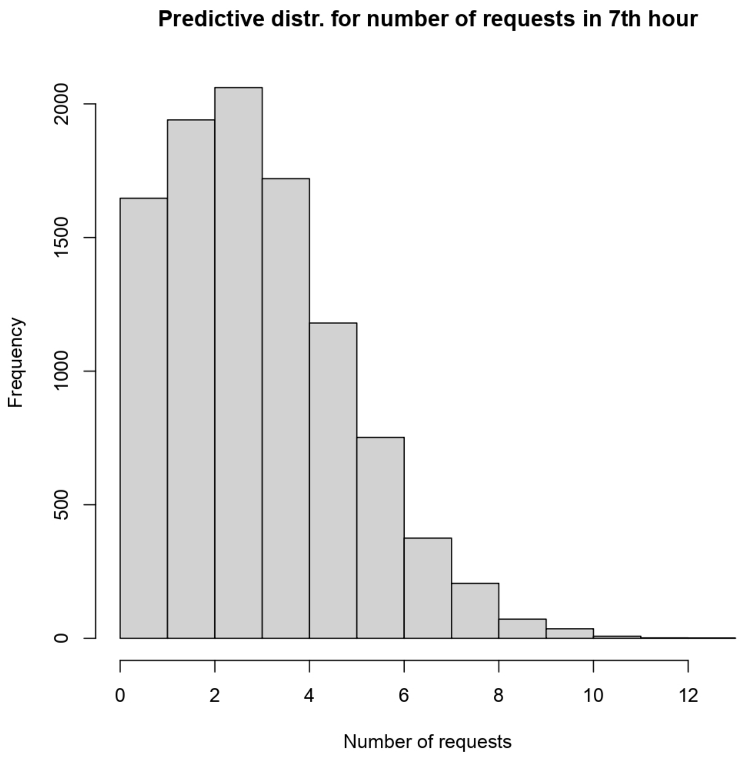

Figure 5: Histogram from the predictive distribution for the number of requests for the 7th hour, sample size of 10 000.

The following code was used to sample the data and plot the histogram shown in figure 5:

```
sampled_lambdas <- sample(lambdas, size=10000, replace
= TRUE, prob=posterior_distr)
integer_sequence <- seq(0, 20)
```

```
requests = list()
for (lambda in sampled_lambdas){
  sample_prob <- dpois(integer_sequence, lambda)
  request_sample = sample(integer_sequence, size=1,
  replace = TRUE, prob=sample_prob)
  requests <- append(requests, request_sample)
}

hist(unlist(requests), xlab="Number of requests",
ylab="Frequency", main="Predictive distr. for number
of requests in 7th hour")
```

2. A game is played on a circular board with numbers 1,2,3,4,5,6 along the edge. Players start at 6. For each round, a 4-sided dice is thrown and the player moves forward the corresponding number of steps. The game is won by a player when the player hits 1 exactly. The only complication is that, whenever you hit 5, you move your piece immediately to 1. Find formulas for the answers to the questions below, and use R to compute the actual values:

(a) Imagine a single player playing the game. What is the expected number of dice throws before the final winning throw?

To calculate the expected number of throws we started by finding the transition matrix.

In the real game there are 6 spots to land on, 1-6, but since landing on 5 is the same as landing on 1 we only model 5 states, states 1-4 and state 6. One state (landing on 6) is the absorbing state, meaning we have won the game. The transition matrix is the following.

$$\text{P} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 0.5 & 0 & 0 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

After that, we partioned the transition matrix and found the fundamental matrix $F = (I - Q)^{-1}$. We could then find the expected number of steps by caclulating $F\mathbf{1}^T$. We start in state 6, and then the probability of going to state 1, 2, 3 and 4 respectively is 0.25 as the dice is 4-sided. By viewing this as the initial distribution we can calculate the expected number of steps by averaging over the entries in $F\mathbf{1}^T$ and adding one for the initial step we ignored.

The expected number of dice throws before the final winning throw is then found to be $7.575758 \approx 8$ throws.

The following code was written to calculate the expected number of dice throws before the final winning throw:

```
tMatrix <- c(0.25, 0.25, 0.25, 0.25, 0,
             0.25, 0, 0.25, 0.25, 0.25,
             0.5, 0, 0, 0.25, 0.25,
             0.5, 0.25, 0, 0, 0.25,
             0, 0, 0, 0, 1)

tMatrix <- matrix(tMatrix, nrow=5, ncol=5, byrow = TRUE)

Q <- tMatrix[1:4, 1:4]
I <- diag(4)
```

```
# Fundamental matrix
F <- solve(I-Q)

ones <- c(1,1,1,1)

E = F%*%ones

avgSteps = sum(E)/4+1
print(avgSteps)
```

(b) What is the probability that the player will visit state 3 before the end of the game?
Either the game ends before state 3 is visited, or it ends after state 3 is visited. Making state 3 an absorbing state and using similar calculations as in *a)*, we can calculate the probability of being absorbed in state 3, i.e. state 3 was visited, or in state 6, i.e. that the game ended before state 3 was visited.

State 3 was made into an absorbing state, and the states were reordered, yielding the following new transition matrix:

$$\mathrm{P} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The probability of visiting state 3 before the end of the game was calculated to be 0.6666667.

The following code was written to calculate the probability that the player will visit state 3 before the end of the game.

```
tMatrix <- c(0.25, 0.25, 0.25, 0.25, 0,
             0.25, 0, 0.25, 0.25, 0.25,
             0.5, 0.25, 0, 0, 0.25,
             0, 0, 0, 1, 0,
             0, 0, 0, 0, 1)

tMatrix <- matrix(tMatrix, nrow=5, ncol=5, byrow = TRUE)

Q <- tMatrix[1:3, 1:3]
I <- diag(3)
R <- tMatrix[1:3, 4:5]

# Fundamental matrix
F <- solve(I-Q)
```

```
FR <- F%*%R

quarters <- c(1/4, 1/4, 1/4)

# prob. of visiting state 3 before the end of the game
prob <- FR[1:3, 1]%*%quarters + 0.25
```

(c) What is the probability that the player will visit state 3 two or more times before the end of the game?
In order to determine if we have visited state 3 two or more times before the end of the game we would like to make an absorbing state for "visiting state 3 a second time". That is not possible in our current state space; by the definition of a Markov chain, we cannot remember if you visited state 3 at some point in the past. Therefore, we must extend the state space of the Markov chain.

Let $T = \{u, v\}$ represent whether state 3 is unvisited or visited at some point in the past. We can then define a new state space $S' = T \times S$. Then, the pair $(v, 3)$, (written as $v3$) can be an absorbing state since that means we have visited state 3 at some point in the past, and we are currently at state 3, meaning that state 3 has been visited twice. From this we can create a 10 by 10 transition matrix similar to the 5 by 5 matrix created before. The difference here is that state $u3$ will always transition to a v state. Intuitively, if you have not visited 3 in the past and are currently on 3, there is a 100% chance that at the next step you have visited state 3 in the past. Other than that, all u-states connect to other u-states and all v-states connect to other v-states. State 6 should still be an absorbing state which means that we get 3 absorbing states in this new Markov Chain: $v3$, $u6$ and $v6$. Our initial distribution will only contain u-states, since we have not visited 3 in the past when there is no past.

With this model we find that the probability of visiting state 3 two or more times is 0.3.

The following code was written to calculate the probability of visiting state 3 twice.

```r
tMatrix <- c(1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
             1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 2, 0, 0, 1, 1,
             2, 1, 0, 0, 1, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 4, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
             0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 4, 0, 0,
             0, 0, 0, 0, 0, 2, 1, 0, 0, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 4)
tMatrix <- matrix(tMatrix, nrow=10,
                  ncol=10, byrow = TRUE)
tMatrix <- tMatrix*0.25

oldOrder <- c(1:10)
newOrder <- c(1:4, 6, 7, 9, 5, 8, 10)
tMatrix[oldOrder, ] <- tMatrix[newOrder, ]
tMatrix[, oldOrder] <- tMatrix[, newOrder]
tMatrix

Q <- tMatrix[1:7, 1:7]
I = diag(7)
R <- tMatrix[1:7, 8:10]

F <- solve(I-Q)
FR <- F%*%R

quarters <- c(1/4, 1/4, 1/4, 1/4, 0, 0, 0)
prob <- FR[1:7, 2]%*%quarters
print(prob) # gives 0.3
```