Stochastic Processes: Assignment 1

Javier Esteban Aragoneses, Mauricio Marcos Fajgenbaun, Danyu Zhang, Daniel Alonso

November 27th, 2020

Importing libraries

```
library(markovchain)
library(matlib)
```

Function to solve the problems

```
# Thanks profe!
matrixpower <- function(M,k) {
  if(dim(M)[1]!=dim(M)[2]) return(print("Error: matrix M is not square"))
  if (k == 0) return(diag(dim(M)[1]))
  if (k == 1) return(M)
  if (k > 1)  return(M %*% matrixpower(M, k-1))
}
```

Problem 1

a)

Markov chain criteria:

1- The probability of being in a state only depends on the previous state.

2- It's a stochastic process.

X = The chain hits state j at time n

X_n is the scenario at time n

All states have finite expected return times and there is only a single communication class so the MC is irreducible, therefore its stationary distribution is **unique**.

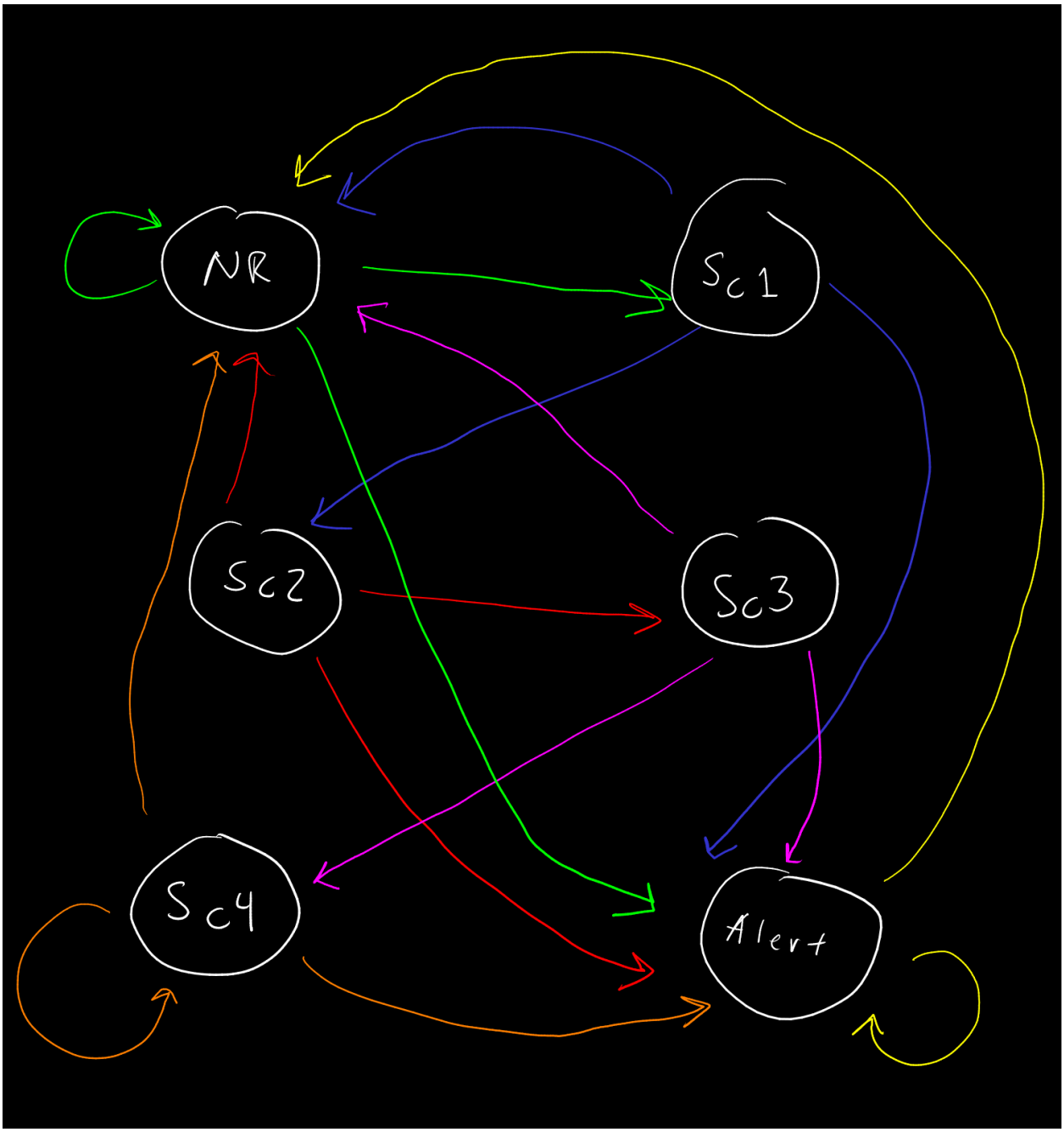


Figure 1: Graph for prob.1

b)

We have first calculated the relative frequencies manually.

```
load('PollutionMadrid.RData')
data <-  X[1,]
mat <- matrix(rep(0,36), nrow=6, byrow=T)
for (i in 1:length(data)) {
  if (data[i] == "Alert") {
    data[i] = 1
  } else if (data[i] == "NR") {
    data[i] = 2
  } else if (data[i] == "Sc1") {
    data[i] = 3
  } else if (data[i] == "Sc2") {
    data[i] = 4
  } else if (data[i] == "Sc3") {
    data[i] = 5
  } else if (data[i] == "Sc4") {
    data[i] = 6
  }
}
data <- as.numeric(data)
for (i in 1:length(data)) {
  mat[data[i],data[i+1]] = mat[data[i],data[i+1]] + 1
}
mat[data[1460],data[1]] = mat[data[1460],data[1]] + 1

tbl <- table(data)
for (i in 1:length(tbl)) {
  mat[i,] = mat[i,]/tbl[i]
}
mat
#>            [,1]      [,2]       [,3]      [,4]      [,5]      [,6]
#> [1,] 0.00000000 1.0000000 0.00000000 0.0000000 0.0000000 0.0000000
#> [2,] 0.00000000 0.9529851 0.04701493 0.0000000 0.0000000 0.0000000
#> [3,] 0.01587302 0.4920635 0.00000000 0.4920635 0.0000000 0.0000000
#> [4,] 0.00000000 0.5806452 0.00000000 0.0000000 0.4193548 0.0000000
#> [5,] 0.23076923 0.3846154 0.00000000 0.0000000 0.0000000 0.3846154
#> [6,] 0.00000000 0.5555556 0.00000000 0.0000000 0.0000000 0.4444444
```

We then tested using the *markovchain* package in order to confirm our results.

```
data <- X[1,]
markovchainFit(data)$estimate
#> MLE Fit
#>  A  6 - dimensional discrete Markov Chain defined by the following states:
#>  Alert, NR, Sc1, Sc2, Sc3, Sc4
#>  The transition matrix  (by rows)  is defined as follows:
#>            Alert        NR        Sc1 Sc2       Sc3       Sc4
#> Alert 0.00000000 1.0000000 0.00000000 0.0 0.0000000 0.0000000
#> NR    0.00000000 0.9529851 0.04701493 0.0 0.0000000 0.0000000
#> Sc1   0.01612903 0.4838710 0.00000000 0.5 0.0000000 0.0000000
#> Sc2   0.00000000 0.5806452 0.00000000 0.0 0.4193548 0.0000000
#> Sc3   0.23076923 0.3846154 0.00000000 0.0 0.0000000 0.3846154
#> Sc4   0.00000000 0.5555556 0.00000000 0.0 0.0000000 0.4444444
```

What can you say of the comparison of your estimates andthe possible transitions between states that you had argued in part a

According to our probabilities shown in the graph. There are 3 arrows with probability 0. This is due to the fact that in the data there are zero transitions from $Sc2 \rightarrow Alert$, $Sc4 \rightarrow Alert$, $NR \rightarrow Alert$, $Alert \rightarrow Alert$.

This is logical given that it is very unlikely to hit an alert state. Unlike the rest of the states.

Later it will be shown that there's a unique stationary distribution (see 1d).

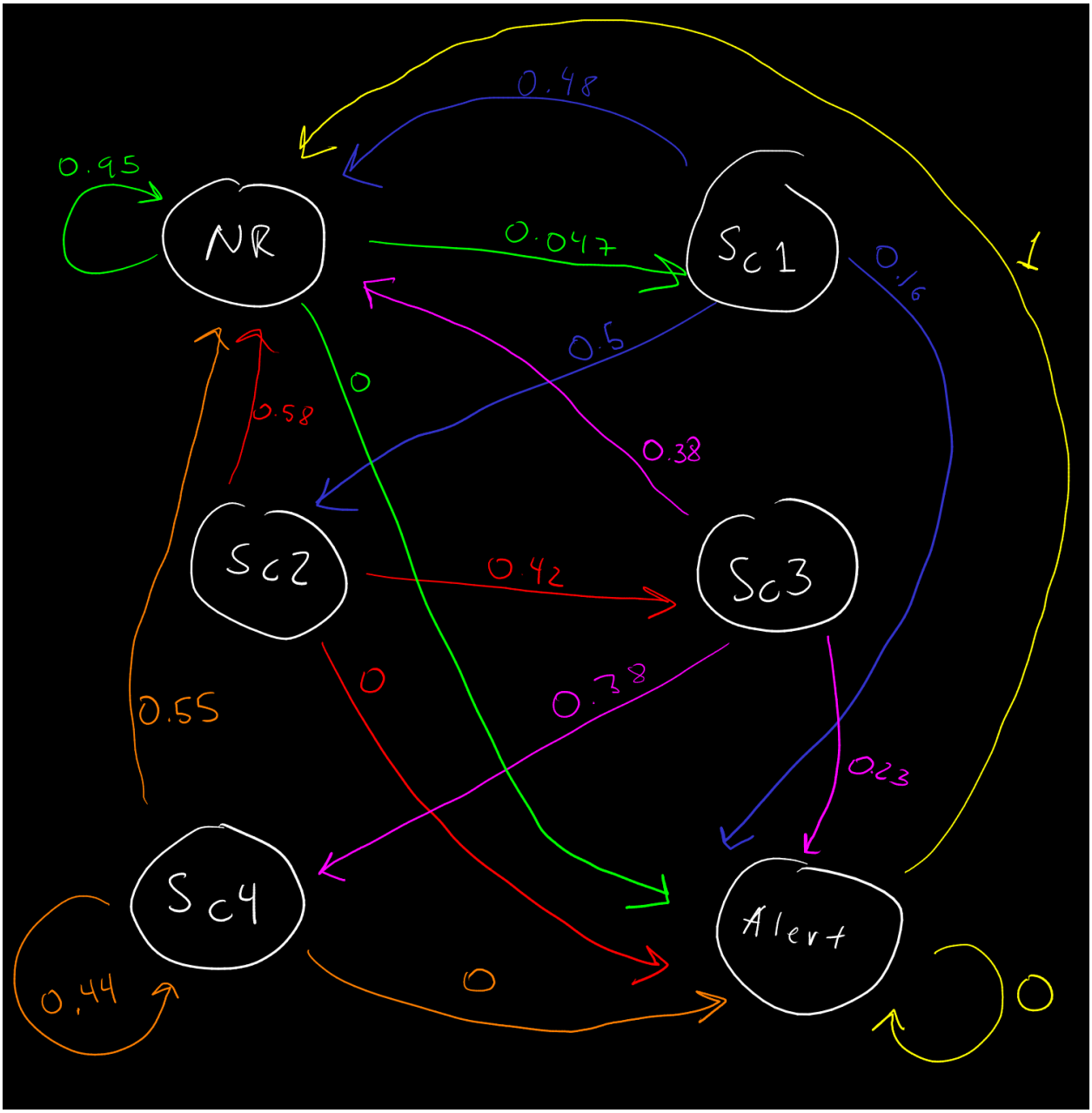


Figure 2: Graph with probabilities for problem 1 (b)

c)

Given that the first state of the chain is NR. We see the following 7 states:

```
data[1:7]
#> [1] "NR" "NR" "NR" "NR" "NR" "NR" "NR"
```

And we calculate the probability as follows:

```
mat[2,2]^7
#> [1] 0.713843
```

We can see the probability is 0.713843

d)

We can see that because we have a unique solution to the system, we have a unique stationary distribution.

```
stationary_dist <- function(P) {
    dim = sqrt(length(P))
    A = P - diag(dim)
    b = matrix(c(1,rep(0,dim-1)),nrow=dim,byrow=T)
    A[,1] <- rep(1,dim)
    print("The solution is the following:")
    return(matlib::Solve(t(A), b))
}
stat_dist <- stationary_dist(mat)
#> [1] "The solution is the following:"
#> x1            =  0.00273973
#>   x2          =  0.91780822
#>     x3        =  0.04315068
#>       x4      =  0.02123288
#>         x5    =  0.00890411
#>           x6  =  0.00616438
stat_dist
#> [1] "x1            =  0.00273973" "  x2          =  0.91780822"
#> [3] "    x3        =  0.04315068" "      x4      =  0.02123288"
#> [5] "        x5    =  0.00890411" "          x6  =  0.00616438"
```

This is our stationary distribution:

$\pi_1 = 0.00273973$

$\pi_2 = 0.91780822$

$\pi_3 = 0.04315068$

$\pi_4 = 0.02123288$

$\pi_5 = 0.00890411$

$\pi_6 = 0.00616438$

Comparing with the proportions we get from our data:

```
rel_error = c()
props = table(data)/length(data)
results <- c(0.00273973,0.91780822,0.04315068,0.02123288,0.00890411,0.00616438)
for (i in 1:length(props)) {
    rel_error[i] <- abs(props[i]-results[i])/results[i]
}
rel_error
#> [1] 1.449998e-06 8.955223e-10 1.142857e-07 1.548387e-07 4.615384e-08
#> [6] 5.7777781e-07
```

We can see our relative errors are all quite low ($<1 * 10^{-5}$) so we could say that the difference between both estimations is nearly negligible.

e)

Proof that this MC has a limiting distribution

We have previously proven that this MC is irreducible. Now we want to prove that this MC is also aperiodic.

We can easily prove this by checking the paths from NR to the nodes it's connected with:

1st path: 3 steps: $NR \rightarrow Sc1 \rightarrow Sc2 \rightarrow NR$)

2nd path: 2 steps: $NR \rightarrow Sc1 \rightarrow NR$)

...

Because the $\{3, 2, ...\}$ and 2 elements of this set are prime numbers, the greatest common divisor of this set is always going to be 1.

This applies for all the other nodes as they all belong to the same class.

Therefore this chain has a **unique** stationary distribution which coincides with its limiting distribution.

What does this mean, in terms of pollution episodes?

The proportion of pollution episodes depends on our limiting distribution, i.e. the long run proportion of pollution episodes with Scenario 1 will be the value of Sc1 in our limiting distribution. If we look ahead in the future, and we want to know the probability of being in a specific pollution episode scenario, p_{ij} is the probability of being in scenario j.

Taking the a high power of our transition matrix we get the following limiting distribution:

```
mp <- matrixpower(mat,120)
mp
#>             [,1]      [,2]       [,3]       [,4]       [,5]        [,6]
#> [1,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
#> [2,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
#> [3,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
#> [4,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
#> [5,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
#> [6,] 0.002739726 0.9178082 0.04315068 0.02123288 0.00890411 0.006164384
```

f)

If we take the sum of the probability of scenarios 3, 4 and Alert: $P(Sc3) + P(Sc4) + P(Alert)$ multiplied by the amount of days in a year, we get the amount of days in a year with such driving restriction:

```
365*(mp[1,1]+mp[1,5]+mp[1,6])
#> [1] 6.5
```

We get that the amount of days where driving is forbidden is 6.5 days. Approximately a week.

Problem 2

a)

We set up the following system of equations:

$$\begin{cases} \sum_{i\in\mathbb{N}\cup\{0\}} \pi_i P_{i,0} = \pi_1 \\ \sum_{i\in\mathbb{N}\cup\{0\}} \pi_i = 1 \\ (1-p)\pi_1 = \pi_2 \\ \dots \\ (1-p)\pi_{n-2} = \pi_{n-1} \\ \dots \end{cases}$$

For the first equation, each $P_{i,0} = p$, therefore:

$\sum_{i\in\mathbb{N}\cup\{0\}} P_{i,0}\pi_i = \pi_1 \Rightarrow p\sum_{i\in\mathbb{N}\cup\{0\}} \pi_i = \pi_1$

And we get:

$p = \pi_1$

For the rest of the equations we have the following:

$(1-p)p = \pi_2, (1-p)^2 p = \pi_3, \dots, (1-p)^{n-1}p = \pi_n, \dots$

Then, we get the following result:

$p + (1-p)p + (1-p)^2 p + \dots + (1-p)^{n-1} + \dots = \sum_{i\in\mathbb{N}\cup\{0\}} \pi_i = 1$

The left side of this equation can be summarized by a summation:

$p\sum_{i\in\mathbb{N}\cup\{0\}}(1-p)^k = \sum_{i\in\mathbb{N}\cup\{0\}} \pi_i = 1$

The summation on the left hand side is a geometric series where $\langle(1-p) < 1$, therefore:

$p * \frac{1}{1-(1-p)} = 1$

This series confirms that $\pi P = \pi$ because:

$$\begin{bmatrix} p & (1-p)p & \dots & (1-p)^n p & \dots \end{bmatrix} \begin{bmatrix} p & 1-p & \dots 0 \\ \vdots & \vdots & \ddots \end{bmatrix} = \pi$$

Performing this matrix product we get:

$$\begin{cases} \pi_1 = p \\ \pi_2 = p(1-p) \\ \pi_3 = p(1-p)^2 \\ \dots \\ \pi_n = p(1-p)^{n-1} \\ \dots \end{cases}$$

Which is the stationary distribution of the chain.

Because our MC is an irreducible infinite state MC, we have a unique stationary distribution π, $\pi_i = \frac{1}{\mu_i}$ and not all states have expected finite return times then we have:

$E[T_i|X_0 = i] = \mu_i = \frac{1}{\pi_i}$

b)

As it is aperiodic and irreducible markov chain, it has a limiting distribution.

c)

We define the following function to simulate n steps with initial value p.

```
sim <-function(x0,n,p){
    n <- n-2
    x = rep(0,n)
    for (i in 1:n) {
        u=runif(1)
        if (u <= p) {
            x[i+1] = x[i] + 1
        } else {
            x[i+1] = 0
        }
    }
    return(c(x0,x))
}
```

We simulate $n = 30$ steps to test the function with initial value $x_0 = 100$ and $p = 0.95$:

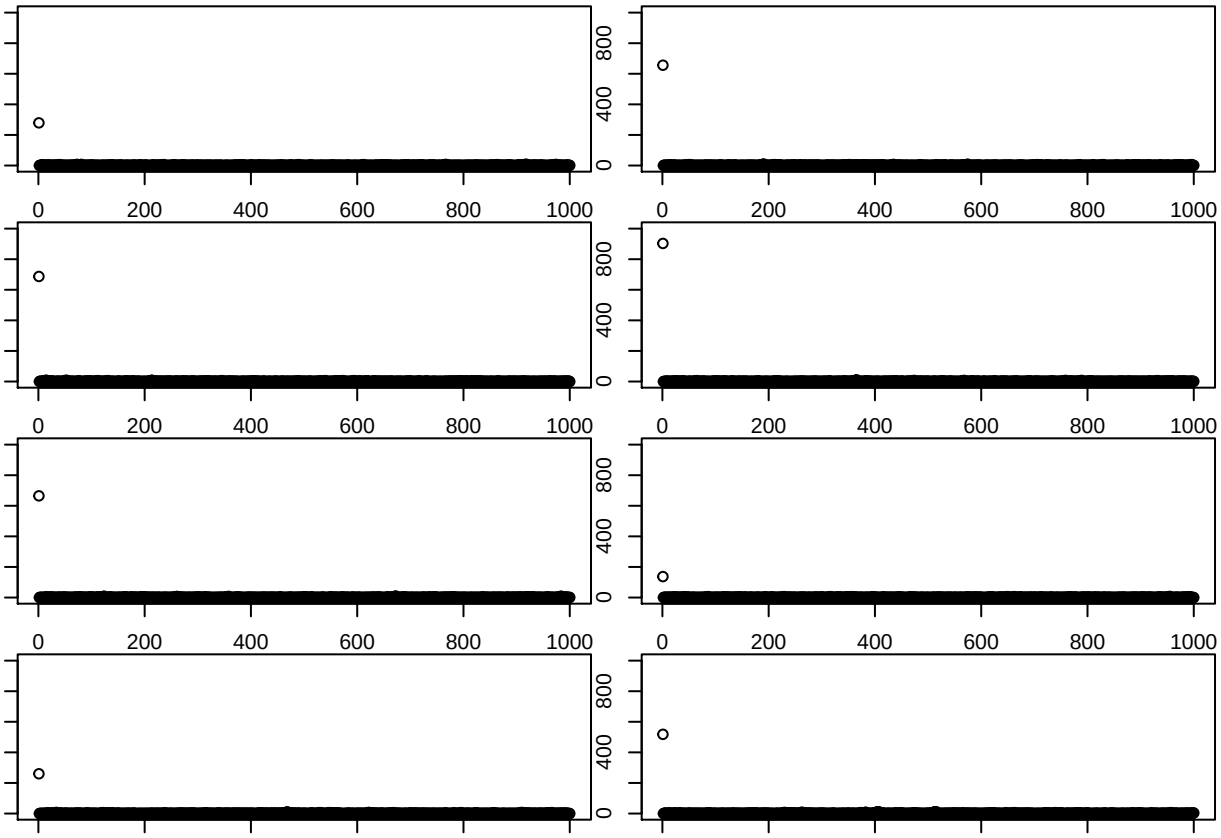
```
sim(100,30,0.95)
#>  [1] 100   0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5   6   7
#> [20]   8   9  10  11  12  13  14  15  16  17  18
```

d)

We generate 8 trajectories of length 1000 with different intiial values between [100,1000]

```
initial_x = sample((100:1000), 8)
trajs = matrix(rep(0,8*1000),nrow=1000,byrow=T)
for (i in 1:8) {
    trajs[,i] <- sim(initial_x[i], 1000, 1/2)
}
```

```
par(mfrow=c(4,2), mar=c(1,1,1,1))
for (i in 1:8) {
    plot(trajs[,i], ylim=c(0,1001))
}
```



The limiting distribution of this chain does not really depend at all on the first value. As such initial value could be a high number (between 100-1000), given that the probability corresponding to our group number ($k = 1$) produces a very high $p = \frac{1}{1+k} = \frac{1}{2}$, therefore the chain drops very quickly to 0 and remains there without climbing values very much.

Our maximums per simulation are always the initial value:

```
maximums = c()
for (i in 1:8) {
    maximums[i] = max(trajs[,i])
}
maximums
#> [1] 279 656 687 903 665 137 260 518
```

While the second highest values will be the followings:

```
maximums = c()
for (i in 1:8) {
    maximums[i] = max(trajs[,i][2:1000])
}
maximums
#> [1]  8  9  8 10  9  7 10 12
```

Which are all significantly lower than our initial values.