Frog crossing the pond puzzle

Pierre-Antoine Roby

2023-01-08

Distribution of the Frog crossing the pond puzzle

On september 12th 2019, Matt Parker of Stand-up Maths youtube channel released a video of a the frog puzzle. The puzzle states that a frog is at one end of a pond and wants to go to the other end. It has an equal chance of landing on every pad separating it from the other side. For example, if there are 9 pads, there is 1/10 chance of getting on any of the pads or on the other side of the pond. The question raised by this puzzle is to predict the average number of jumps the frog will have to take before getting to the other side of the pond. after every jumps, it still has an equal chance of landing on every pad separating it from the other shore or on the shore itself. From the last example, if from the first jump it landed on the third pad, it will have 1/7 chance of landing on every of the remaining pads.

From the video, the challenge was solely focused on the average jumps needed to go though the pond. As other probability distributions are a lot more complex than just a mean. I was wondering if we could express the distribution graphically like we usually do for the binomial distribution for example. How can we calculate the probability of going on the other side with an n number of jumps?

The objective of the document is to resume how it can be calculated using probability matrix in a markov chain.

For simplicity, the other edge of the pond will be referred to as a last and final pad.

Link to the introductory video by Matt Parker: https://www.youtube.com/watch?v=ZLTyX4zL2Fc&ab_channel=Stand-upMaths

Probabilty matrix of the distribution

If we think of probability matrix for this kind of distribution, it is basically a table which resume for every starting pad, what is the possible landing pad. In the table below the starting pads are the columns and the landing pads are the rows. The example for the introduction is kept using 10 pads but it could be any number of pads. For n number of pads the matrix should be of $n + 1$ since the starting platform need to be taken into account.

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    0    0    0    0    0    0    0    0    0     0     0
##  [2,] 1/10    0    0    0    0    0    0    0    0     0     0
##  [3,] 1/10  1/9    0    0    0    0    0    0    0     0     0
##  [4,] 1/10  1/9  1/8    0    0    0    0    0    0     0     0
##  [5,] 1/10  1/9  1/8  1/7    0    0    0    0    0     0     0
##  [6,] 1/10  1/9  1/8  1/7  1/6    0    0    0    0     0     0
##  [7,] 1/10  1/9  1/8  1/7  1/6  1/5    0    0    0     0     0
##  [8,] 1/10  1/9  1/8  1/7  1/6  1/5  1/4    0    0     0     0
##  [9,] 1/10  1/9  1/8  1/7  1/6  1/5  1/4  1/3    0     0     0
## [10,] 1/10  1/9  1/8  1/7  1/6  1/5  1/4  1/3  1/2     0     0
## [11,] 1/10  1/9  1/8  1/7  1/6  1/5  1/4  1/3  1/2     1     1
```

The last 1 from the 10th pad (index [11,11]) is there to calculate the cumulative distribution. For a probability distribution, it can be replaced with a zero. This probability matrix will be called P.

State matrix and Markov chain

A markov chain is meant to shows evolution in a certain system. It is usually used to evaluate what would be the equilibrium but it is not in our interest since the equilibrium is acheved when the frog is on the other shore and stays there. Nothing interesting there, but if we use it to check on every state, it is the same as getting the probability of the frog being on a specific pad for every pads. The state matrix at the start is:

```
##       [,1]
##  [1,]    1
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
```

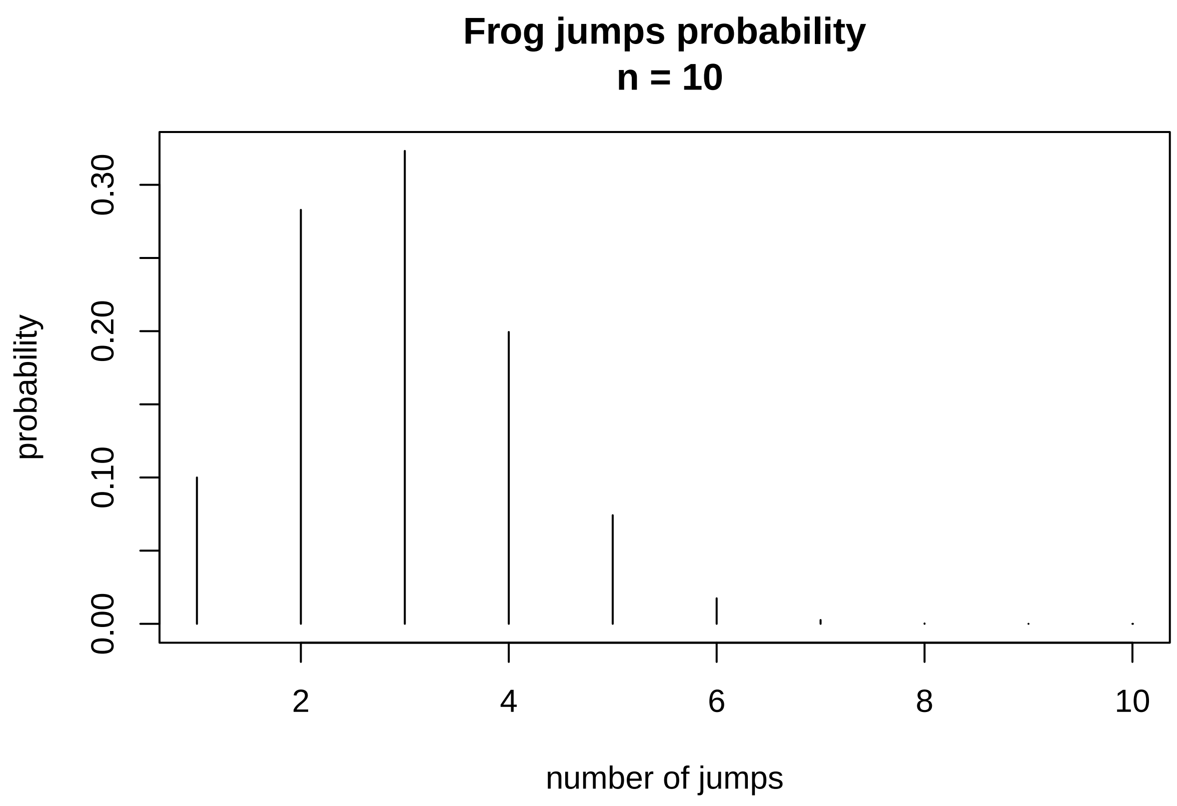
The one at the start signify that the probability of being on the starting shore is certain and on the other pads is null. The states will be called X_n with the one shown higher as X_0.

If we use a Markov chain, we can update the probability at every jump. Each state being a jump. This way, if we multiply the probability with the any state we obtain the next state. $PX_0 = X_1$ $PX_1 = X_2$ and so on...

If we extract the probability on index [11,1] of the n^{th} state variable, we obtain the chance that the frog is on the other shore on the n^{th}. For example, here are the state matrix for 5 pads.

```
##
## [1] "X1"
##       [,1]
##  [1,]  0.0
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
## [11,]  0.1
##
## [1] "X2"
##             [,1]
##  [1,] 0.00000000
##  [2,] 0.00000000
##  [3,] 0.01111111
##  [4,] 0.02361111
##  [5,] 0.03789683
##  [6,] 0.05456349
##  [7,] 0.07456349
##  [8,] 0.09956349
##  [9,] 0.13289683
## [10,] 0.18289683
## [11,] 0.38289683
##
## [1] "X3"
##              [,1]
##  [1,] 0.000000000
##  [2,] 0.000000000
##  [3,] 0.000000000
##  [4,] 0.001388889
##  [5,] 0.004761905
##  [6,] 0.011078042
##  [7,] 0.021990741
##  [8,] 0.040631614
##  [9,] 0.073819444
## [10,] 0.140267857
## [11,] 0.706061508
##
## [1] "X4"
##               [,1]
##  [1,] 0.0000000000
##  [2,] 0.0000000000
##  [3,] 0.0000000000
##  [4,] 0.0000000000
##  [5,] 0.0001984127
##  [6,] 0.0009920635
##  [7,] 0.0032076720
##  [8,] 0.0087053571
##  [9,] 0.0222492284
## [10,] 0.0591589506
## [11,] 0.9054883157
##
## [1] "X5"
##               [,1]
##  [1,] 0.000000e+00
##  [2,] 0.000000e+00
##  [3,] 0.000000e+00
##  [4,] 0.000000e+00
##  [5,] 0.000000e+00
##  [6,] 3.306878e-05
##  [7,] 2.314815e-04
##  [8,] 1.033399e-03
##  [9,] 3.935185e-03
## [10,] 1.505980e-02
## [11,] 9.797071e-01
##
## [1] "X6"
##               [,1]
##  [1,] 0.000000e+00
##  [2,] 0.000000e+00
##  [3,] 0.000000e+00
##  [4,] 0.000000e+00
##  [5,] 0.000000e+00
##  [6,] 0.000000e+00
##  [7,] 6.613757e-06
##  [8,] 6.448413e-05
##  [9,] 4.089506e-04
## [10,] 2.376543e-03
## [11,] 9.971434e-01
##
## [1] "X7"
##               [,1]
##  [1,] 0.000000e+00
##  [2,] 0.000000e+00
##  [3,] 0.000000e+00
##  [4,] 0.000000e+00
##  [5,] 0.000000e+00
##  [6,] 0.000000e+00
##  [7,] 0.000000e+00
##  [8,] 1.653439e-06
##  [9,] 2.314815e-05
## [10,] 2.276235e-04
## [11,] 9.997476e-01
##
## [1] "X8"
##               [,1]
##  [1,] 0.000000e+00
##  [2,] 0.000000e+00
##  [3,] 0.000000e+00
##  [4,] 0.000000e+00
##  [5,] 0.000000e+00
##  [6,] 0.000000e+00
##  [7,] 0.000000e+00
##  [8,] 0.000000e+00
##  [9,] 5.511464e-07
## [10,] 1.212522e-05
## [11,] 9.999873e-01
##
## [1] "X9"
##               [,1]
##  [1,] 0.000000e+00
##  [2,] 0.000000e+00
##  [3,] 0.000000e+00
##  [4,] 0.000000e+00
##  [5,] 0.000000e+00
##  [6,] 0.000000e+00
##  [7,] 0.000000e+00
##  [8,] 0.000000e+00
##  [9,] 0.000000e+00
## [10,] 2.755732e-07
## [11,] 9.999997e-01
##
## [1] "X10"
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    1
```

The above calculations shows the output matrices for every state. The last term of every state is the cumulative distribution of chances the frog is on the other shore by the n^{th} jump. Here it is in a probability graph.



The probability of 8, 9 and 10 jumps are so low that they are barely visible. The process can be done with every number of pads. From those numbers classic probability can be done to get mean, variance and other statistical values.

Here is the code used to run the calculations in R.

```r
# for nbPad the other side of the pond is counted as a pad
getProbDistributionFrog <- function(nbPad) {
  # Setting up the matrix of probability for each jump
  mat <- matrix(data = 0, nrow = nbPad + 1, ncol = nbPad + 1)
  for(a in 1:nbPad+1){
    for(b in 1:nbPad){
      if(a>b){
        mat[a,b] <- 1/(nbPad+1-b)
      }
    }
  }
  mat[nbPad+1,nbPad+1] <- 1

  # Setting up the matrix for the initial state
  etat <- matrix(data = 0, nrow = nbPad + 1, ncol = 1)
  etat[1,1] <- 1

  # Result statistic vector
  cumul <- c(1:nbPad)

  # Creates the cumulative vector of the distribution
  for(i in 1:nbPad){
    etat <- mat%*%etat
    cumul[i] <- etat[nbPad+1,1]
  }

  # Probability vector
  prob <- cumul

  for(j in 2:length(prob)){
    prob[j] <- prob[j] - cumul[j-1]
  }

  # Calculates the mean of the distribution
  frogMean <- 0
  for(k in 1:length(prob)){
    frogMean <- frogMean + (k)*prob[k]
  }

  plot(c(1:length(prob)),prob, type = "h", xlab = "number of jumps",
       ylab = "probability", main = paste("Frog jumps probability\n n =", nbPad))

  return(data.frame(matrix(c(cumul,prob,frogMean, rep(NA,nbPad-1)),
       nrow = 3, ncol = nbPad, byrow = TRUE),
       row.names = c("cumulative", "probability", "mean")))
}
```