Numerical Recursion Project

Matthew Hefner

December 10, 2018

We wish to design an algorithm that approximates the probability of the survival of a dynasty and, by extension, the annexation of their country, within 10 generations. Annexation occurs if the dynasty is left with no male heir. To find this, the algorithm below approximates the probability of the survival of the dynasty for 10 generations using n trials given some probabilities of having heirs. We can then find the probability of annexation using the fact that $1 - P(Survival) = P(Annexation)$. We then wish to then compare these approximations when we change heir probabilites via investing in fertility treatment. A recursion diagram is given on the last page to help follow the code below.

```
import random
import sys
random.seed(20) # So I can discuss results
sys.setrecursionlimit(10000)
def generation(dads, gen, succ, trial, n):
  if trial == n: # First trial is trial 0
    return succ / n # Probability is returned on last trial
  if gen == 10: # Successful trial
    return generation(1, 0, succ + 1, trial + 1, n)
  sons = 0
  for j in range(dads):
    x = random.uniform(0.0, 1.0)
    if x <= 6/11:
      sons += 1
    elif x <= 7/11:
      sons += 2
  if sons == 0: # Unsuccessful trial
    return generation(1, 0, succ, trial + 1, n)
  return generation(sons, gen + 1, succ, trial, n)
def kzovck(n):
  #1 - P(survival) = P(annexation)
  print("Probability Approximation with", n, "trials:", generation(1, 0, 0, 0, n))

kzovck(200)
```

```
## Probability Approximation with 200 trials: 0.02
```

```
kzovck(500)
```

```
## Probability Approximation with 500 trials: 0.026
```

```
kzovck(1000)
```

```
## Probability Approximation with 1000 trials: 0.03
```

```
def generation(dads, gen, succ, trial, n): # Redefine with treatment
  if trial == n: # First trial is trial 0
    return succ / n # Probability is returned on last trial
  if gen == 10: # Successful trial
    return generation(1, 0, succ + 1, trial + 1, n)
  sons = 0
  for j in range(dads):
```

```
    x = random.uniform(0.0, 1.0)
    if x <= 6/11:
      sons += 1
    elif x <= 8/11:
      sons += 2
  if sons == 0: # Unsuccessful trial
    return generation(1, 0, succ, trial + 1, n)
  return generation(sons, gen + 1, succ, trial, n)
#With treatment:
kzovck(200)
```

```
## Probability Approximation with 200 trials: 0.135
```

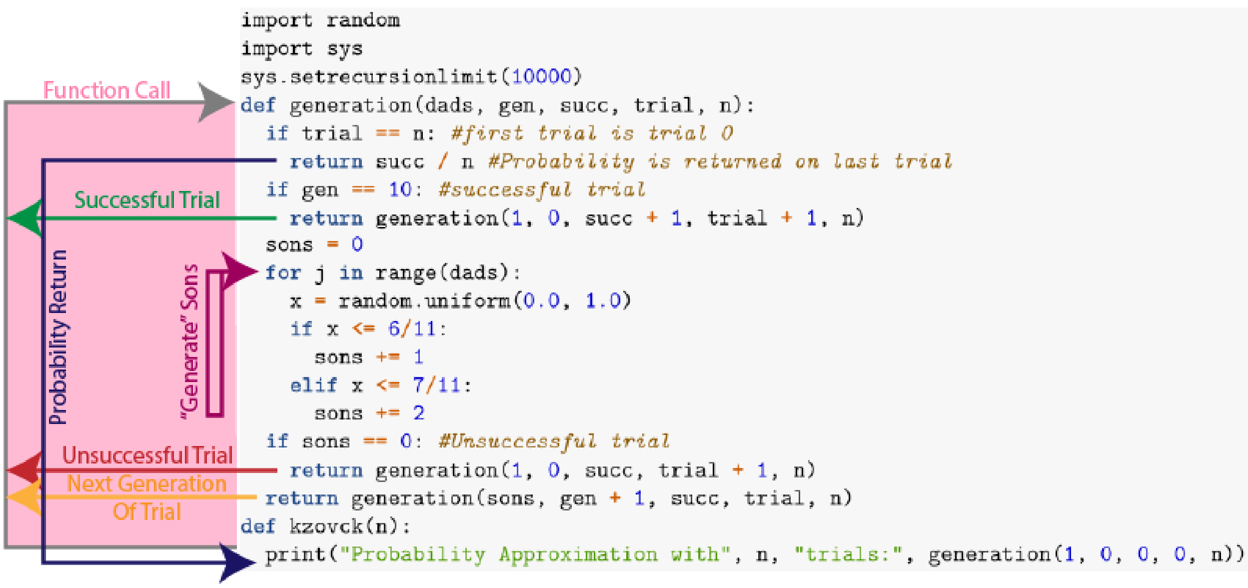
```
kzovck(500)
```

```
## Probability Approximation with 500 trials: 0.17
```

```
kzovck(1000)
```

```
## Probability Approximation with 1000 trials: 0.181
```

Based on these approximations, the probability of *survival* increased relatively dramatically from around $0.02 - 0.03$ ($2\% - 3\%$) to approximately $0.135 - 0.181$ ($13.5\% - 18.1\%$), but there is still clearly a low probability of survival to 10 generations. Since $1 - P(Survival) = P(Annexation)$, this means the probability of *annexation* within 10 generations decreased from $0.97 - 0.98$ ($97\% - 98\%$) to approximately $0.819 - 0.865$ ($81.9\% - 86.5\%$). Therefore it is evident that it is worthwhile to invest in the treatment, but it is still fairly likely that they will not survive and the country will be annexed within 10 generations.




```
import random
import sys
sys.setrecursionlimit(10000)
def generation(dads, gen, succ, trial, n):
  if trial == n: #first trial is trial 0
    return succ / n #Probability is returned on last trial
  if gen == 10: #successful trial
    return generation(1, 0, succ + 1, trial + 1, n)
  sons = 0
  for j in range(dads):
    x = random.uniform(0.0, 1.0)
    if x <= 6/11:
      sons += 1
    elif x <= 7/11:
      sons += 2
  if sons == 0: #Unsuccessful trial
    return generation(1, 0, succ, trial + 1, n)
  return generation(sons, gen + 1, succ, trial, n)
def kzovck(n):
  print("Probability Approximation with", n, "trials:", generation(1, 0, 0, 0, n))
```

Figure 1: Recursion Diagram