Practical21 Conditional probabilities sample sol

Simon

10/04/2022

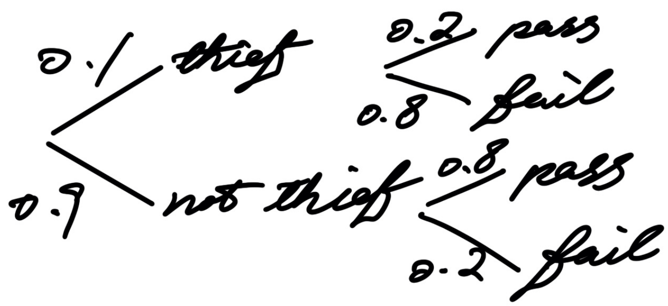
Lie detector problem

1. By setting up and running a simulation

```
t=0;i=0
case=list()
while(t<50){
 staff=sample(c('Thief','NotT'),1,prob=c(0.1,0.9))
 if (staff=='Thief') {
  result=sample(c('Fail','Pass'),1,prob=c(0.8,0.2))
  x=ifelse(result=='Fail',1,0)
 }
 else {
  result=sample(c('Pass','Fail'),1,prob=c(0.8,0.2))
  x=ifelse(result=='Fail',1,0)
 }
 t=t+x;i=i+1
 case[[i]]=data.frame(Staff=staff,Result=result)
}
TrueT=bind_rows(case) %>% filter(Staff=='Thief',Result=='Fail') %>% nrow
```

So 13 of the staff are thieves

2. By using Bayes' theorem



$P(\text{thief}) = 0.1$, $P(\text{fail}) = 0.1 \times 0.8 + 0.9 \times 0.2 = 0.26$

$$P(\text{fail} \mid \text{thief}) = P(\text{fail} \cap \text{thief}) / P(\text{thief})$$
$$= 0.8 \times 0.1 / 0.1 = 0.8$$

$$P(\text{thief} \mid \text{fail}) = P(\text{fail} \mid \text{thief}) P(\text{thief}) / P(\text{fail})$$
$$= 0.8 \times 0.1 / 0.26 = 0.31$$

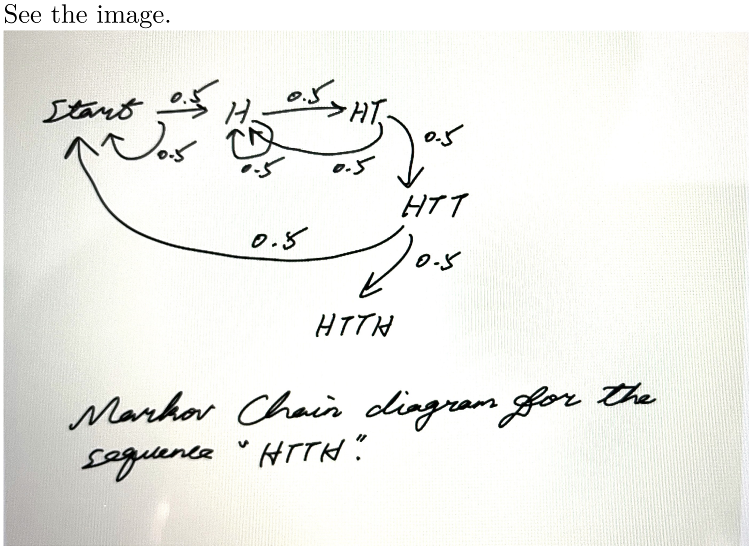
$\#$ of thieves $= 50 \times 0.31 \approx 15$

.

Coin toss

1. Complete the Markov chain diagram we started in the lecture

See the image.



2. Using the Markov chain, create a simulation and run it many times to answer your question

Approach 1:

```r
N=c()
for (n in 1:1000){
t=0;i=0
while(t<1){
 # sample the first letter until a head shows
 repeat{
  start=sample(c('H','T'),1)
  i=i+1
  if(start=='H') break
  }
 # sample the second letter until 'HT' shows
 # here we already have the initial state 'H'
 # so the sampling only goes around first state
 repeat{
  second=paste(start,sample(c('H','T'),1),sep='')
  i=i+1
  if(second=='HT') break
  }

 # sample the third letter until 'HTT' shows
 # the sampling will start again from the first state if we have 'HTH'
 repeat{
  third=paste(second,sample(c('H','T'),1),sep='')
  i=i+1

  if(third=='HTT') break

  else{
   repeat{
    second=paste(start,sample(c('H','T'),1),sep='')
    i=i+1
    if(second=='HT') break}
  }
  }

 # sample the forth letter, t is the condition that I used to
 # stop the while loop once we have the first 'HTTH'
 # the sampling will start from the initial state if we have 'HTTT'
 # so like repeating the previous states over and over
 repeat{
  forth=paste(third,sample(c('H','T'),1),sep='')
  i=i+1

  if(forth=='HTTH'){
   t=t+1
   break
   }

  else{
   repeat{
```

```r
    start=sample(c('H','T'),1)
    i=i+1
    if(start=='H') break
    }
   repeat{
    second=paste(start,sample(c('H','T'),1),sep='')
    i=i+1
    if(second=='HT') break
    }
   repeat{
    third=paste(second,sample(c('H','T'),1),sep='')
    i=i+1
    if(third=='HTT') break
    else{
     repeat{
      second=paste(start,sample(c('H','T'),1),sep='')
      i=i+1
      if(second=='HT') break}}}
  }
  }
}
N[n]=i
}
avg=mean(N)
```

It took us 17.958 iterations on average to generate the pattern.

Approach 2:

```r
#function to extract matrices
extractMatrices <- function(mcObj) {
require(matlab)
mcObj <- canonicForm(object = mcObj)
#get the indices of transient and absorbing
transIdx <- which(states(mcObj) %in% transientStates(mcObj))
absIdx <- which(states(mcObj) %in% absorbingStates(mcObj))
#get the Q, R and I matrices
Q <- as.matrix(mcObj@transitionMatrix[transIdx,transIdx])
R <- as.matrix(mcObj@transitionMatrix[transIdx,absIdx])
I <- as.matrix(mcObj@transitionMatrix[absIdx, absIdx])
#get the fundamental matrix
N <- solve(eye(size(Q)) - Q)
#computing final absorbion probabilities
NR <- N %*% R
#return
out <- list(
canonicalForm = mcObj,
Q = Q,
R = R,
I = I,
N=N,
NR=NR
)
return(out)
}
```

```r
M=matrix(0,nrow=5,ncol=5,dimnames=list(c('Start','H','HT','HTT','HTTH'),
               c('Start','H','HT','HTT','HTTH')))
M[1,1:2]=0.5;M[2,2:3]=0.5;M[3,c(2,4)]=0.5;M[4,c(1,5)]=0.5;M[5,5]=1
MMC=as(M,'markovchain')
Result=extractMatrices(MMC)
xixi=Result$N %>% rowSums %>% sum
```

It took 58 steps to reach the final state.
Ref: https://journal.r-project.org/archive/2017/RJ-2017-036/RJ-2017-036.pdf, page 9-11.