# ADS2 - Conditional probabilities, model solution

Dmytro Shytikov (based on Simon's solution)

Semester 2, 2023/24

These solutions are just model ones. Thus, you might have developed your own strategy that gives comparable results but uses more optimal code. It is absolutely fine as soon as our results are comparable.

## Lie detector problem

### Calculations

The probability of a random employee being a thief is $P_{thief} = 0.1$. Thus, the probability of a random employee *not* being a thief is $P_{(not\ thief)} = 0.9$.

The probability of the detector giving a correct result (a thief is identified as a thief and an honest employee is identified as an honest employee) is $P_{(correct\ result)} = 0.8$. Thus, the probability of getting the wrong result of the detector check (false negative for a thief and false positive for an honest employee) is $P_{(wrong\ result)} = 0.2$.

50 employees gave positive results (are 'hits') and they blame the result of their check being wrong. Who would be upset because of the result of the check? Obviously, an honest employee (because he/she did not steal anything; thus, the result is false positive for the honest employee) or a thief (because he/she stole something and may get caught; thus, the result is correct for the thieves).

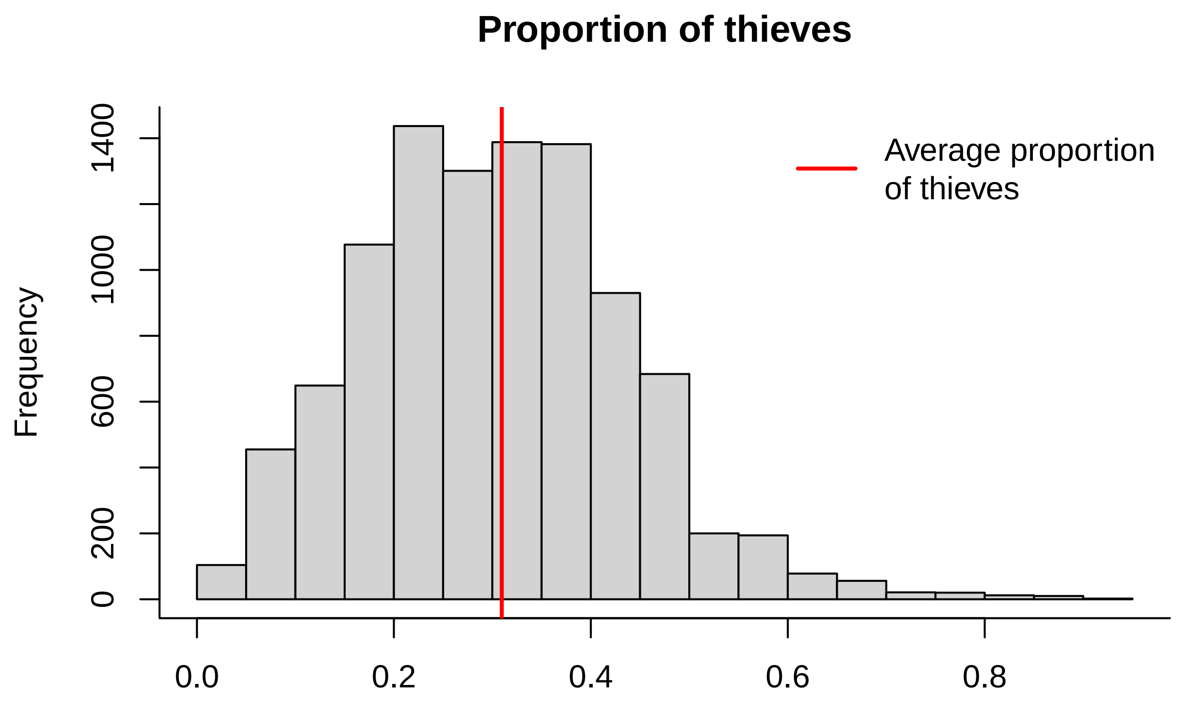
The probability that the 'hit' is an innocent fellow is $P = 0.9 * 0.2 = 0.18$. The 'hit' may be a thief with the probability of $P = 0.1 * 0.8 = 0.08$.

Thus, our sample space decreases to the sum of these two categories because we are interested only in those who are wrongly blamed for honest persons or true crimes. So, the probability that a person from this sample (two categories and 50 persons) is a thief in $P_{(thief\ AND\ correct\ conclusion)} = 0.08/0.26 = 0.3$. Thus, out of 50 persons, 15 are most probably crimes.

## Simulations

We want to create a sample of 50 persons that includes innocent persons with a probability of 90% (thus, the probability of getting a thief is 10%). Then, we identify whether this person passes the lie detector check or fails it (thus, it is either claimed to be an honest person or a thief) with the respective probability for each category that is unequal (thieves fail checks in 80% chance and honest person fail checks in 20% chance).

```r
# I will use the `replicate` command to generate a sample but you can use another way
# I set the sample size of 50 for the sake of simplicity. But you may set your number.
  cases <- replicate(n = 50, expr = {
    Staff = sample(x = c("Innocent", "Thief"), size = 1, prob = c(0.9, 0.1))
    if(Staff == "Innocent"){
      checkResult <- sample(x = c("Pass", "Fail"), size = 1, prob = c(0.8, 0.2))
    }else{
      checkResult <- sample(x = c("Pass", "Fail"), size = 1, prob = c(0.2, 0.8))
    }
# Please, note that the probability of failing for an honest person and a thief
# is unequal!
    Results <- c(Staff, checkResult)
    return(Results)
  }) %>% t() %>% as.data.frame()
# Now, you have a sample of employee. You just need to extract and summarize information
# from it in any useful and convenient way you prefer and repeat it many times.
# I have used ~10000 loops.
```



## Coin toss

Somebody tosses a fair coin repeatedly, and records the sequence of outcomes (e.g. “H-T-T-T-T-H-H-T-. . .”). How long would it take until the sequence “H-T-T-H” first appears?

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

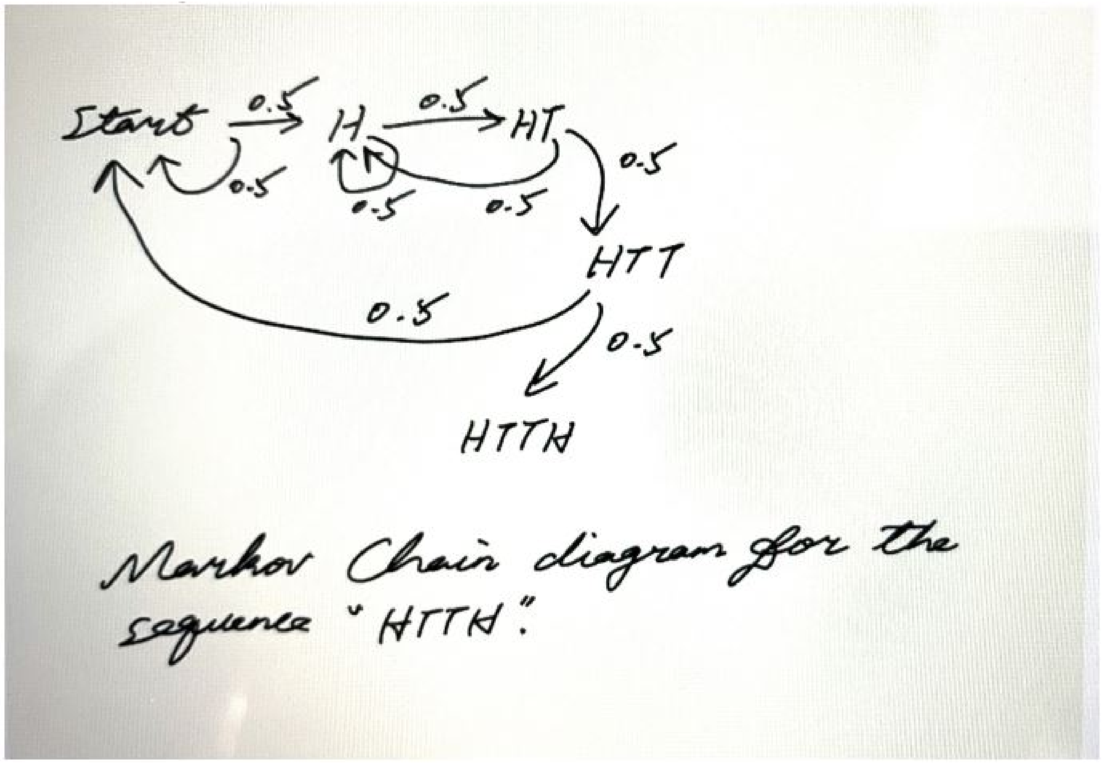


Figure 1: Markov chain for the HTTH sequence, by Jingyuan Chen

## Simulation

This code is just for the general idea. You may create your own more interesting and better code.

```r
# If we go from the O state to T instead of H, we need to restart the cycle.
# If I get H, I continue.

sequenceWanted <- c("H", "T", "T", "H") %>% paste(collapse = "")
STEP = 1
sequencecurrent <- c(sample(x = c("H", "T"), # I start the sequence
                            size = 1,
                            prob = c(0.5, 0.5)))# Determine the first toss
print(sequencecurrent) # These steps are not essential. I just want to see the progress

# Then, I start a code that stops only when my current sequence equals the wanted one.
while(sequencecurrent != sequenceWanted){
  if(nchar(sequencecurrent) == 1 & sequencecurrent != "H"){# Proofread the first toss
    # And wait till I get H
    while(sequencecurrent != "H"){
      STEP <- STEP + 1
      sequencecurrent <- sample(x = c("H", "T"), size = 1, prob = c(0.5, 0.5))
      print(sequencecurrent)
    }
  }else{ # Otherwise, go to the second step
    STEP <- STEP + 1
    sequencecurrent <- c(sequencecurrent,
                         sample(x = c("H", "T"),
                                size = 1,
                                prob = c(0.5, 0.5))) %>%
      paste(collapse = "")
    print(sequencecurrent)
  }
  if(sequencecurrent != "HT"){ # Check the second step and proofread it
    #STEP <- STEP + 1
    sequencecurrent <- substr(sequencecurrent,
                              nchar(sequencecurrent),
                              nchar(sequencecurrent))
    print(sequencecurrent)
  }else{ # Otherwise, proceed to the third step
    STEP <- STEP + 1
    sequencecurrent <- c(sequencecurrent,
                         sample(x = c("H", "T"),
                                size = 1,
                                prob = c(0.5, 0.5))) %>%
      paste(collapse = "")
    print(sequencecurrent)
  }
  if(nchar(sequencecurrent) == 3 & sequencecurrent != "HTT"){# proofread the third step
    sequencecurrent <- substr(sequencecurrent,
                              nchar(sequencecurrent),
                              nchar(sequencecurrent))
    #STEP <- STEP + 1
    print(sequencecurrent)
  }else{ # otherwise, go to the fourth step
```

```
    STEP <- STEP + 1
    sequencecurrent <- c(sequencecurrent,
                         sample(x = c("H", "T"),
                                size = 1,
                                prob = c(0.5, 0.5))) %>%
      paste(collapse = "")
    print(sequencecurrent)
  }
  if(sequencecurrent != "HTTH"){ # restart the whole chain
    sequencecurrent <- substr(sequencecurrent,
                              nchar(sequencecurrent),
                              nchar(sequencecurrent))
    #STEP <- STEP + 1
    print(sequencecurrent)
  }else{ # otherwise, finish the whole sequence
    print(sequencecurrent)
    print("Sequence complete")
  }
}
STEP
```

---