Experiments, Conditional Probability, and Bayes' Theorem

Hadijat Oke 117818512 Thomas Urdinola 117971262

Contents

Simulating Experiments

In this problem we will simulate two important experiments: coin tosses and die rolls.

We will explore empirical probability of events using random samples. The two important functions to understand are "sample" and "replicate".

We use the sample function to set up our experiment, and replicate function to repeat it as many times as we want.

Die Rolls

Setup the experiment where we roll two fair six 6-sided die independently, and calculate the sum of the numbers that appear for each of them.

```
roll_dice <- function(){
  die1 <- sample(1:6, 1, replace = TRUE)
  die2 <- sample(1:6, 1, replace = TRUE)
  return(die1 + die2)
}

roll_dice()
```

```
## [1] 8
```

Use the replicate function to repeat this experiment 10000 times, store the output of the experiment in a variable "sum_die".

```
sum_die <- replicate(10000,roll_dice())
sum_die
```

```
##     [1]  8  8  7  6  6  5  7  4  6  3  8  3  6  5  6  6  9  5  5 10  5  7 10  5
##    [25]  6  8  2  3 11  7  6  4  8  6  8  4  7  8  9  5  2  5  6  6  4  7  7  8
##    [49]  8  7  9  9  9  8  2  7 10  6 11  3  3  8 12 10  9  8 11  6  7  8  3  5
##    [73]  7  8 11  4  8  5 11  3  7  9  7  3  5  4  6  6  9  7  9  5  5  9  7  4
##    [97]  9 10 11  6  9  3  8  2  6  7  7  8  8  7  4  5  4  9  9 10  4  9  6  8
##   [121]  6  7  7  9  2  5  4  6  9  5  6  8  5  8  3  3  2 10  8  8  6  8  3  6
##   [145]  5  9  9  4  5  5  6  7 12  6  7  5  6  4  8  6  5  3  9  6  8  6  4 11
##   [169]  5  8  6  6  9  8  9 10  8  5 12  6  6 10  5  6 10  7  6  6  8  4  6  4
##   [193] 10 10  4  7  6  8 10 10  9  7  9 11  8 10  7  7 12  6  8  7  9 11  3  6
##   [217] 10  4 12  6 10  4  5  6  7  5  5  6  8  8  4  6  3 11  4  9  9  9  6  4
##   [241]  7  7  6  5  7 11  4  5  5  8  9  6 12  3 10  5  3 10 12 11 10  4  4  4
##   [265]  9  5  3 10  7  8  9  9  7  6  6 10  6  8  4  8  2  6  3  8 12  7  7  7
##   [289]  6  9  6  7  7  5  6  7  7  4  9  2  7  6  4  6 11  4  6  6  9  9  6  8
##   [313] 10  2 12  7 11 10  5 10  8  5  9 10  7  3  8 10  3  8  3  8  9  6  7  7
##   [337]  8  3 10  8  9  9  7  4  4  7  9  9  6  6  9  4  4  4  6  6  9 10  8  7
##   [361]  5  4  9  5  2  4  6  5  9  5  5 10  4  9  8  3  7  5  8  8  8  2 10  8
##   [385] 10  2  3  5  4  3  7  8  6  8  3  6  8  9  8  6  8 12  7  5  3  5  3  7
##   [409]  7  5  6  5 10  4  8  5  9  5 10  5  8  8  5 10  7 11  7  8  9  7  5  9
##   [433]  8  5  3  6  5  5  5  4 10  7  5  3  8  5  7  4  8  6  8 10  5  7 10  7
##   [457]  7 11 10  8  5  5  6  8  9  6  8  8  7  4  3  6  8  8 10  2  9  9  4  9
##   [481]  6 10  7  6  7 10 11  7  2  9  6  4  9  6  6  5  7  8  6  7  7  6  7  2
##   [505]  7  8  9  4  9  6  6  6  7  6  2  8  7 10 10  3 10  8 10 10 10  8  7  7
##   [529] 10  2 10  3  4  2 11 12  9  7  6  7  4  4  4  7  6  8  3  7  9  8  7  4
##   [553]  6  4  8  6  6  7  4  3  2  5 10  9  9 11  6  6  7  3  4  7  9  3  8  7
##   [577]  7  3  4  8  5  8  9  9  8  8  6 10  7  8  8  4  4 12  8  5  8  6 10 10
##   [601]  5 11  9  8  8  5  5  7 10  7 11 11  6  5  7 10  9 11 10  4  7  7  7  8
##   [625]  7 10  9  4  5  9  3  6 10  4  7 11  8  8  4 12  7  9  5  5  5 11 10  8
##   [649]  9  7  8  6 10  8  8 10 12  6  9  7  7 10  4  6 11 12  7 12  7  9 10  7
##   [673]  9 11  7  4  5  5  8  9 11  9  6  6  9  7  3 12  6  6  8  8  4  4  8  7
##   [697]  8  5  6  3  9  7  3 10  6  9  4 10  4  2  9  6  4  5  4  8  7  9  2  5
##   [721]  4  8  9  8  5  6 12  6  5 11  5  9  8  6  5  5 11  4  7  5  7  7  4  3
##   [745]  4  8  5  4  6 11  7  9  5  6  5  6  7  9  7  3  4  9  7  7  5 11  8  6
##   [769]  4  5 10  7  4  5  6  5 11  7  5  8  6  8 10  8  4  5 11  6  6  9 11  5
##   [793] 10  8  6  7  6  7  8  6  5  9  6  8  9  7  7  5  6  6  7 10  5 10  6  5
##   [817]  5  9  4  8  6  9  3  7  8  8  4  6  4 11  7  8  5  3  4  7  8  6  6  4
##   [841]  3  4  4  8  2  8  3  2 11  9  9  6  7  7 11  5  7  8  9 12  5  7  6  7
```

```
##   [865]  8  9 10 10  8  8  9  6 10  6  8  4  8  9 10 10  3  9  4 10 11  7  6  7
##   [889]  3  5  8  7  6  6 12 10  7  4 10  6 11 12  6 10  6  8  3 11  8  2  7 10
##   [913]  7  6  2 11  4  5  9  6  3  2  9  9  9  7  9 10  7  6  4 10  6  4  9  8
##   [937]  7  4 12 12 10  8  5  2  7  2  9 11  8  7 10  6  7  5  8  3  3  9  8  7
##   [961]  6  5  6  5  7  7 11  8  6  6  5  8  8  9  3  5  4 12  7  8 11  6 10  9
##   [985] 11 10  6  8  5  8  8  8  5 11  9  9  6  8 12  7  6  9  6  5  4  7  2  7
##  [1009]  3 11  7 10  6  4 10  6  3  6  2  6  7  7  7  6  6  8  6  7  2  5  4  6
##  [1033]  4  9  5  6  3  9 10  8  9  6  8 10  8  6  8  6  6  6  9  5  8  8  8  7
##  [1057]  3  6  8  5 10  4  8  7  9  7  4  4  5  9  7  7  6  4 12  8  8  4  9 11
##  [1081]  7  3  8  4  2  7  7  6  9  3  5  4  7  9  8  6  9  7  4  7  4  8 11  7
##  [1105] 10  4  8 12  4  6  3 11  7 11  3  8  8  6 10  5  9 10 10  8  7  7 11 11
##  [1129]  5  3  6  7  5  6  3 10  9  4  6 12  5  5 10  7  6  8  4  7  4  6  8  7
##  [1153]  5  7  9  8  8 10 11  4  7  3  9  9  9  8  9 12  5 10  4  7  8  6  6 10
##  [1177]  6  7  8  7  7  8  4  4 10  5  8  8  5  8  3  6  5  7  8 11 10  6 11  2
##  [1201]  6  5  2  3  8  6 12  4  3  4  5  5  8  4  7  4  8  8  7  6  4 12  4  3
##  [1225] 12  8  2  5  7  7 12  8  7  3 11 10  4  5 10  9 10  4  7 11  8  7  8  6
##  [1249]  2  6  3  7  4  9 10  9 11  8 10 11  5  4 10  7  8  4  5  5  7  7  7  7
##  [1273]  6  8  6  5  6  5  3  7  4  6  2 12  9  5  5  8  4  8  7  5  9  5  6  7
##  [1297]  2 10  5  5  7  7  6  4 12  4 11  5  6  5  8  3  8  8  9  4  6 11 11  5
##  [1321]  5  3  2  8  7  8  9  4  9 10  3  4  6  7  7  7  6  6  3  4  9  9  7  7
##  [1345]  6  9  8 10 10 10  9  7 12  8  5  5 11  5  8  4  4  9 10  8  4  6  5 10
##  [1369]  6  9  3  7  6  7  7  5 10  8  7  8  5  3  6  7  8  3  4  4  8  9 10  4
##  [1393]  8  4  6 11  9  4  3  7  9  8  8  5  4  6  9  8  7  8  9  2 10  7  5  8
##  [1417] 12  7  8 10  7  4  8  9 10  6  7  7  7  8  2  9  6  9  6  4  5  6  6  7
##  [1441]  7  4  9  9  6  7  4 12  4  8  8 11  4  5  8  9  5  6  5  8  7 11  6  8
##  [1465] 10  8  5  6  7  5  9 10  6 12 11  9  6  9  6  2 11  8  3  7 10  7  8  4
##  [1489]  4  5  9 11  5  6  5 10  5  6  7  9  5 10  2  6  3  5  6  5  9  7  9  5
##  [1513]  7  6  9  8  2 10  4  6 10  7  8  6 12  4  8  4  7  8 10  7  7  6  8  6
##  [1537]  9 10 10  7  5  5  5  5  6  9  5  7  4  7  3  4  3  2  5  6 11  7  6 10
##  [1561]  6  7  8  7 12  4  9  2  6  7 10  9 11  9  5  4  8 10  6  3  5  5 10 10
##  [1585]  6  9  5  9  5  6  9  9  9  7  6  9  7  8  9 12  8  8 11  6  5 11  4  6
##  [1609]  6 10  7  9  8 10  8 10  7  6  8  3 11  8  6  5  4  8  5  4  7 11 10  4
##  [1633] 10 10 12 10  7  9  6  5  6  3  6  9  9  7  6 10  7 10  8  6  6  2  9  6
##  [1657]  8  7  7  5  6  4  6  9 12  4  8  7  3  7 12  5  8  4  6  5 11  7  7  8
##  [1681]  7 12  5  5 10  9  6  9  4  9  8  3  3  8  4  5  5  7 11  3  5  8 10  7
##  [1705]  9  9 10 10  9  6 10  9  7  6  6  4  5 11  5  4  8  4 12  8  5  5  8  4
##  [1729]  6  6  7  5  9  3  2  7  7  7  8 10  6  9  5  5  9  2  8 10  5 11  8  3
##  [1753]  8  7 10  7  7 11  6  8  4 10  7 10  6  4  4  9 11 10 11  8  6 11  6 10
##  [1777] 11  7  9  8  5  2  8  6  8 10  4  9  4  9  9  7  8 12 10 11 10  7  5  9
##  [1801]  5  5 11  6  8  3  6  4  9 11  7 10  7 11  5  6  4  9  8  8  8  3  3  5
##  [1825]  8  9  5  8  7  8 10  8 10  7  7  8  7 10  9  9  5  6  8  8 12  7 10  6
##  [1849]  9  8  5  8  2 10  7  9  4  9  5  8  7  7  8  3  6 11 10  5 10  8 10  7
##  [1873]  4  5  8  6  8  4  8  5 11  4  7  8  9  6  7  5 10  5  5  7  4  3  3  8
##  [1897]  7  9 10 12  5  5  8 10 12  8  6  4  6  9  2  6  6 11  6 10 11  5  8 11
##  [1921] 11  4  7  7  5  4  7  2 11  6  7  6 12  8  8  7 10  9  9  9  3  9  3  5
##  [1945]  7  4  8  8  7  8  7  8  5  6  3  8  5  4  5  2  5  8 10  5 11  3  9  4
##  [1969]  5 10  7 10  8  7  6  7  6  5  9  8 10  7  5 10  5  4  4 11  3  7  9  6
##  [1993]  7  5  8  4  3  8  8  5  3  5  6  3  5  5 11  7  9 12 10  8  6 12  8  3
##  [2017]  7 10  9  5  5  5  8  4  4 10  4  5  6  6  5  9  6  4 11  2 10  5  7  5
##  [2041]  2  7 10  5  8  4  7  6 12  6  7  5 12  7  8  6 12  8  7  5 11  4  5  8
##  [2065]  5  7  8 12  7  6 10 11  4 10  7  7  9  5 10 12  6  6  3  7  8  6 11 10
##  [2089]  6  9  4  4  3  5  4  6 10  6 10  6  9 10  7  8 10  4  4  2  4  4  3  5
##  [2113] 11  7 12 10  6  2 10  6  6  6 12 12  7  6  4  7  7 10  8 10  6  7  3  3
##  [2137]  8  7  7  3  7  6  9  7  9  7  4  7 10  7  2  6  7  5  5  3  9  5 10  3
```

```
##  [2161]  4 10  5 10  7 10  8  6  5 10  9 11  7  4  7  9  8  7  6  8  3  9  8  6
##  [2185]  9  8  4  9  7  9  8  5 10  5  5  4  6  4  5 11  4  8  8  2 10  5  8  7
##  [2209]  7 10  4 11  9 10  5  5  8  7 10  7  9  8  6  3  4  7  7  6 10  7  7 10
##  [2233]  6  5 11  2  8  9  8  9  7  4 10  5  6  4  8  9  9  8  9  7  7  6 12  2
##  [2257]  7 11  5  5  8  8  6 11  5 11  6  7  7  6  8 12  4  9  7  8  8  6  8  6
##  [2281]  7  5  6  6  7  7 11  6  3  5 12  6  8  4  6  5  6  6  6  9  3  7 10  8
##  [2305]  9  3  7 11  4  9  7  7  6  6  6  7 10  6  7  6  6  2  4  4  9  2  7  5
##  [2329]  9  8 10 12 12  7  5  5  6  6  7  7  2  6  3 10  9  6  4  4  7  7  7  8
##  [2353] 11  9  6 11  7  9  7  7 10  9  4  3  6  2  6  8  4 11  5  5  3  3  7 10
##  [2377]  4  8 12  3  4 10  6  6  6  7  9 11  4  2  3  3 10  2  9  5  7  7  7  9
##  [2401]  8  7  8  6  5  3 11  7  6 11  6  6  8  4  6 11 11  3  4 10  4  4  5  9
##  [2425] 10 12  8  4  5  5  4  4 11  3  5  6  4  6  6  8  7  3  6  6  5  8  6  7
##  [2449]  7  8 10  9  7  2  5 10  6  5  4  7 11 12  5  9  3 11  5  7  9  6  3  8
##  [2473]  8 11 12  6  3  6  5  7  7  4  7  6  5 10  6  5  4  8  5  5  7  8  6  8
##  [2497]  6  4  6  9  6  8  7  6  6  3  7  6 10  3 10  4  6  3  2  9  9  6  7  7
##  [2521]  6 11 11 11  4  4  9  4  7  6  4  6  6  5  8  4  7  6  7 10  4  7  9  9
##  [2545]  7  9  9  8  6  6  6  8  8  5  4  8  5  4  8 11  8 10 11  8  7  9  9  7
##  [2569]  8  4  9  8  5  8  6  8  8  8 10  2  5  4  4  8  5 11  6  8  7  8  7  5
##  [2593]  4  3  5  5 10  7  7  6  5  5  4  4  5  6  3  5  4  7  7  8  5 10  3  8
##  [2617] 11  3  9  7  7  7 10  7  6  6  8  9  7 10  7 10  5  6  7  4 10 10  6  8
##  [2641]  5  6  2  7  5  8  8  3 11  6  9  7  5 10  5  4  7  8  7  8  6 10 11  7
##  [2665]  7  4  8 11  6  3  8  5 11  4  3  9  9  8 11  9 10  3  7  7  9  8  4  8
##  [2689]  7  7  5  7  7  7  6  8 11 11 10 12  9  6  6  7  8  7  6  9  2 11  4  4
##  [2713]  3  5  5  9  5  9  2  5  6  8  6  5  6  8  5  6  6  6  4  7  9  7  9  6
##  [2737]  8  9  4 10  7  9 11  7  8  4  7  4 11  6  7  2  8  3  9  7 11  8  5  5
##  [2761]  9  4  4  6  9  5  7 11 10  8  2  9  3  7 10  7 11  8  6  5  7  8  4 10
##  [2785]  7  7  6  3 10  6  9  5  9  8 10  6  7  8 11  9  8  6  5  6 10  8  7  6
##  [2809]  9 10  3  5  3  6  5 10  8  8 11  7 10  4 10  7  8  6  8  9  5  9  6  5
##  [2833]  5  9 11 10  7  5  4  9 10  8  9  8 10  4  7  9  7 10  7  9  5  4  9  7
##  [2857]  4 10  7 10  9  8  6  4 11  7  9  7  6  7  5  9  6  9  4  6  9  5  2 12
##  [2881]  5  7 10  2  3 10  7 12 11 10  2  6  3  9 10  7  7  6  3  8  8 11  8  8
##  [2905] 10  6 10  6  6 10  4  8  5  7 10  6  5 11  9  7  5 11  2  7  5 11  6  8
##  [2929]  4 10  7 10  5  6  6  9  7  9  8  7  8  8  9  4  3  3  5  5  4  6 10  9
##  [2953]  8  6 12  7  6  9  7  9  6  6  9 11  2  8  9  8  7  6  4  9  7  5  8  7
##  [2977]  7 10 10  7  2  9  7  6  2  7  6  3  7  4  5 11  6  3  6 11  8 10 10  3
##  [3001]  5 11  4  9  8  2  6  5  4  5  4  7  8 10  8  7  9  9  6  5  5  6  7  7
##  [3025]  2  5 11  8  7  6  7  9  2  5  9  7  3  4  9  9  9  6  5  9 10  4  2  3
##  [3049]  6  6  6 11  6  8  6  7  5 11 10  6  4  7  5  7 12  6  6  3  4 11  3  5
##  [3073]  8  6  6 10  3  6  7  3  6  7  8  4  7  5 11  8  3  3  4  6  6  7  7  9
##  [3097]  6  4  7 11  7  8  6 11  7  7  3  6  6  4  8  9  5  2  5  6  7  9  6  9
##  [3121]  8  9  8  7  7 10  6  5  3  5  7  4 10  8  6  9  5  9  6  4  5 10  6  5
##  [3145]  6  9  5  8  7  2  6  7  9  5  6  7  5  4  7  9 11  8  3  8  4  6  2  8
##  [3169]  8  9  5  9  9  4  8  7  5  9  3  7  9  5  9  5  7  8  7  8  8  9  8  8
##  [3193]  6  9  5  5  9  9  5  9  2  7  8  6  6  7  2  6  3  8  8  6  9  6 12 11
##  [3217]  2  8  4  4 10  2 12  6  4 10  3  6  6 12 12  9  3 12  5  6  7  5  9  9
##  [3241]  7  9  9  6  7  3  7  7 12  5  7  6  5  5  6  8 10  8 12  8  7  8  7  9
##  [3265]  6  8  5 11  6  7 10  6  9 10  3  5 10  4  9  5  5  4 10 12  6  4  7  9
##  [3289]  3  9  8  6  5 10 11 10  7  7  6  3  9  5 10  8 11  7  9  7  4  7  5  5
##  [3313]  7  4  3 10  9 10  7  5  5  4  4  8  3  7  7  5  6  7  5  6  6  7  7  3
##  [3337]  5 10  4  4 12  4  6  4  4 11  6  6  3  6  6  6 10  2  8  7 11  7  6  9
##  [3361]  8 11 11  4  7  9  7  8  4  6  5  5 10  5  4  6  4  8  8  5  8  7  2  8
##  [3385]  4  8  8  6  5 10  7  6 10  3  7  5  9 11 11  7  4  6  4  3  9  8  7  6
##  [3409] 10 11  7  7  4  3  8 10  4 10  6 11  6  8  8  6 11  2  7  7  5  4 12  9
##  [3433]  7  7  3  8  8  5  9  7  6  4  5  8  9  6  8  6  5  8  7  2 12  9 11  4
```

```
##  [3457]  6  7 11  7  7  5  9 10  7  6  8  8 10  8  5  9  9 10  5  8  7  6  9 10
##  [3481] 12  5  6 11  8  5  5  6  8  4  8  8  9  7  6  8 11  6  8 11  8  6  5  8
##  [3505]  8  7 10  8  3 12  4  3  5  4  7  6 11  8  8 11  7  9 10  4 10  8  4  6
##  [3529]  7  7  6  8  7 11  8  7  9  6 10  2  7  4 11 11 12 11  5  9  6  5 11  4
##  [3553]  5  8  4 10  5  5  7  6  3  8  8  9  6  9  9  7  5  9  6  9 12  8 11  9
##  [3577]  3  5  5  6  8  6  3  7  3 12  9  7  6  6 11 11  3  6  7  6  8  9  4  8
##  [3601]  3  7 11  7  6  9  7  4  5  5  4  2  7 10 11  9  9  9  3  9  4 10  8  4
##  [3625]  6  9  5  6  7  7  6  6  9  4 12  3  8  5  8  7  4  7  5  9  6  5  3 10
##  [3649]  9  7  5  7  6 11  7  9 11 12  5 10  3  7  7  8  8 11  9  8  8  6  4 11
##  [3673]  4  7  4  8 11  7  6 10  8  9  9  9  9  3  8  9  4  9  9  7  5  9  9  8
##  [3697] 10  3  3  4 10  6  9  6  8  5  3  9  8  8  4  2  5  8 10  4 10  4 10  5
##  [3721]  8  5 10  8  8  5  6  6  7  6  4  9  9  6  5 11 12  8  9  6  2  7  8  7
##  [3745]  7  4  6  8  7  3  9  3  2  6  8 10  7  6  6 10  7  5  8  4 11  8  6  6
##  [3769]  7  6  8  9  4  6  6  8 11  2  6  7  5  7 10  4  7  6 10  9  8  9  7  5
##  [3793]  6  7  7  9  8 10  9 11  8  8 10 10  7  9 10  4  8  3  5  9  3  5  6  9
##  [3817]  5  5  7  2 12  5  4  5  9 11  5  5  7 11  8  9  6  6  6  6  7  3  5  8
##  [3841]  7  8  4  5  7  7  3  2  9  7  8 12  5  5  8  3 11 10  7  4 10 11  5  5
##  [3865]  7  7  8  9  7 11 10  7  4  6 11  8  7  8  3  7  3  9  6  5  7  7  5  7
##  [3889]  5  5  7  3  8 11  9 11  3  9  7  8  8  8 11 10  7  8 11  3  7  3  9  7
##  [3913]  9  6 10  7  4  7  7  5  6  9 10  4  5  7  4 12  2  8  7  5 11  3  6  6
##  [3937]  5  5 10  7  8  7  4  4  6  7  3  3  9  7  7  7  8  9 10  9  7  8  6  6
##  [3961]  5  4  6 10  7  7 10  4  7  8  7  9  8  5  3  9  8  7 11  3  9 12  5  7
##  [3985]  9  7 10  7  4  8  7  4 10  6  8  5  5 10  5  5  7  3  2  9  8  7  6 10
##  [4009]  5  7 10  5  9  6  9  8  9  9  8 10  7  9  9  5  7 11 11 12  9  6  9  6
##  [4033] 12  9  8  3 11 11  6  9  9  7  9 10  5  9  6  7 10  6  7  8  6  8  3  2
##  [4057]  9  8 10  9  6  3  8  8 11  5 11  4 10  7  5  5  8  9  6  9  5  5  2 10
##  [4081]  7 12  5  9  7  7  4  5  6  2  2  8  5 12  9  7  9  8  3  7  6 10  3 11
##  [4105]  6  4  6  5  4  9 11  7  9  6  8  9  6  5  8  6  8  4  8  4  3  9  8  6
##  [4129]  9  5  9  7  7 11  7  8  7  7  7  5  8 12  6  7  5  6  8  4  8  8 11  7
##  [4153]  7  8  7  7  8  7  6  4  9  6 10  7  7  7  6  5 11  4 10 10 10  4  4  3
##  [4177]  6  4 12  7  3  8  8  7  7  4 11 10  9  6  7  7  5  8  9  3  6  3  6  2
##  [4201] 11 11 10 11 11  8  5  3  4  8  9  8  5  8  7  5  7  5  5  4 12  5  8  6
##  [4225]  7  6 10  3  8 11 10  7 10  3  7  7  7  7  6  9 11  3 10  2  6  7 10  5
##  [4249]  7 12  4  6 10  9 12  6  6  4  8  8  7  2  6  8  5 10 12  2 11 11  4  9
##  [4273]  7  2  9  5  9  9 10  7  9  9  8  4  6 10  8 12  4 10  7  5  7  7  5  6
##  [4297]  9 10  8  9  8  6  8  6  7  5  9  6  7  9  4  9  7  7  6 11  6  4  6  8
##  [4321] 10  3 10  5  8  3  7  2  7 10  7 12  8  8  6  7  7 10  8  7  6  4 11  9
##  [4345]  8  7  4 10 10 10  8  9 10  5  7  8  2  6  4  7  3  9  9  9 11  7 10  8
##  [4369] 10  9  3  8  3  7  8  8  7  6 10  8  8  4  4  6  8  9  5  5  5  5  7  6
##  [4393]  5  6 12 10  7  9 12  4  6  3 10  6  9  7  5  5  8 10  9  4  8  8 11  4
##  [4417] 10  5  8  7  8  5  3  8  2  9  2  5  9 11  7  9  8  8  7  6  2  3  6  7
##  [4441]  5 12  8  3  3 11 11  9  7  8  9 12 10  8  4  8  9  4  6 11  2 11  8  3
##  [4465]  6  2  3  9  8  8 11  7  8  3  7  7  6  8 11  8 10  5  5  8  9 10  8  7
##  [4489]  5 11  4  4 11  4  5  6  3  5  3 10  7 12  6  4  4  7  7  3  7  6  9  7
##  [4513]  8 10  9  4  5  8  6  7  5  4  7  9  6  9 12  4  7  6  8  6  4 12  5  6
##  [4537]  7 10  7  9  6  7  7 11  7  7 12  9  9  3  7  4  8  6  9 10  4  4  4  6
##  [4561]  8  9  7  4 10  2  9 11  8  7  5  7  7 10  7  6  3  4  4 11 10  9 10  6
##  [4585]  6  9  6  8  6  6  7  9  4 11  6  8  8  5  3  7  5  8 11  4  8  5  5  8
##  [4609]  7  4 10  7  8  5  6  7  8  3 11  7  7 10  8  9  9  9  4  6  5  8  5  9
##  [4633]  8  7  7  7  5  7  3 12  4  7 10  4  9  3  3 10  8  8  6  6  4  9  3  9
##  [4657]  5  5  3  9  4  6 11  8  9 10  8  9  7  4  9  9 12  4  2  7  4  6  4  9
##  [4681]  7 11  4  3  8  7 10  7  4  4  5  6  7  6  8  5  7  7 11  5 11  4  4  6
##  [4705] 11  4  8  7  7  8  8  6  6  3  9  8  5 11  8  5  7  4  5  6  5 11  4  7
##  [4729]  4  8  5  3  8  7  4  9 11  7 10  3  5  4  8  6  6  3  9  4  7  3  3  8
```

```
##  [4753]  9  8  5  9  9  7  5 12 11  8  7  7 10  3 12  4 11  8  3  4  9  6 10  5
##  [4777]  4 11  6  5  6  3  9  9  5  7  4  8  4 10  8  8  5  3  7  8  3  7  5  3
##  [4801] 11  7  4  3  7  6  7  6  7  4  9  9 11  6  6  5  8  4  8  6 10 11 10  2
##  [4825]  8  9  4  6  7  4  8  4  7  4  3  6 11 10  4  5  6  8  8  8  9  6  8  9
##  [4849]  7  5  9  9  6  4  9  9  6 10  6 10  6 10  7 12 11 10  4  4  5  8  8  9
##  [4873]  9  5  6  6  5  5  6  4  6  8  8  7  9  9  6 11  8  7  6  4  6  7  5 10
##  [4897]  7  8  6  9 11 10 10  7  8  7  9  9  7  3 11  8 10  3 11  4  7  5  3  3
##  [4921]  5  8  7  4  7  3  5  5 10  2  8  5  9  7  6  7  7 11  8  7  9  3  6  4
##  [4945]  8 11  7  8  3 10  7  8  8 10  4  7  6  7  7  2  9  5  3 12  9  6  8 10
##  [4969]  8  2  5  6 12  6  8  8  5  2  7  7  5  5  9  6  3  5  8  8  9 12  9  6
##  [4993]  4  9  7  6  7 11  9  4  5  6  8  8 11  8  4  4  9  9  5  8 11  7  7 11
##  [5017]  9 11  3  9  9  7  5  9  8  5  7  8  3  8  7  6 10  6 10  5  8 11 11  2
##  [5041] 10  4  8  3  7  6  8  2 11 11 10 10  7  4  6  9  6  4 11  7  8  6 10  9
##  [5065]  5  7  5  7  7  6  9  4  7  6  2  6  6  7  9  7  9  4 10  5  7 10  7  3
##  [5089] 10  9 10  9  2  9  3 11  8  9  8  7  6 10  7  8 11 11  7  2  3  9  6  3
##  [5113] 11  9  6  5  7  9  9 11 10  4  9 10  3  5  7  3  3 11  8  6  2  4  3 10
##  [5137] 11  9  5  8  8 11  3 11  8  7  8  7 10  2  7 10  7  6  7 10  6  9  8  3
##  [5161]  7  9  8  8  8  9  7  9  8  6  8 11  8  5  6  8 10  5  4  4 10  6  8  5
##  [5185]  4  7 11  9  6  3  8  9  3  8  6  7  7  6  9  4 10  6  7  6  6  8  6 10
##  [5209]  6  6 10 10  7  7  9  4  8  4 10  6  8  6  8  9 10 12  7  6  8 10  4 10
##  [5233]  8  9  7  7  8  5 10 11  5  7 11  7  5  2  7 11  8  8  8  5  7 12  8  7
##  [5257]  7  5  7  7  7  5 11  8  4  5 11  9  8  6 11 11  9  4  4 10  7  5  5  9
##  [5281]  5  6  8  6  7  7  7  5  5  6  8  4  7  6 11 11  4  6  7  8  7  4 11 10
##  [5305] 11  8  4  7  3 10 12  5  5  7  6  7  5 11  7  3  8  6  7  9  8 12  5  3
##  [5329]  7  7  6  3  5  8  6  4  4  6  7  8  8  8  7  6  5  5  8  7  7  7  4  8
##  [5353]  5  5 11  7  6  2  3  4  9 10  4  8  5  7  6  9  5 11 10 10  5  6  8  5
##  [5377]  7  7  3  2 11  5  5  9 11  9  5  6  6  4  7  5  5  9  8  6  3  6  2  2
##  [5401]  7  4  9  4  7 11  9  3  6  7  8  6  9 12 10  6  8  7 10  4  7  5  7  7
##  [5425]  7  8  2  9  7 11 10  8  9  9  8  4  8  9  6  6  6  2  4  2  9  7  5  3
##  [5449]  3  8  9  7  3  7  5  3  8  6  7  5  4  2 11  9  6  7 10 10  6  7  5  6
##  [5473] 12  3  9  9  7  3  7  9  7  5  7  4  6  7  7  4 10  3 11  6  8 10  7  6
##  [5497] 10  4  3  6 10  4  7  4  5  7  3  8  4  5 10  9  8  6  5  8  9  7 10 12
##  [5521]  4  4  4 12  6  6  5  6  9  4  7  6  4  3  5 10  6  7  7  3  8  4  4  5
##  [5545]  9  7  6  2  9  8  7  9  4  7  3  7  8  5  6  6  4 10  8 10  5  7  3 10
##  [5569]  6  5  3  8 10  9  4  7  5  7  8  3  8  4  8  4  4  5 10  7  6  7  7  5
##  [5593]  9  2 11  6  9 11  6  9 10  8  8  9  7  2 10  5  5 12  6  4  9  6  4  7
##  [5617]  5  5  7  2 12  6  4  7 10  9  9  5  8  8  7  5  9 10  7  5  7  5  8  6
##  [5641]  4  5 10  4  2  7  5 10  6 10  7  4  5  7  8  4  9  9  6 11 12  5 10  6
##  [5665]  4  5  2  4  9  3 12 10 11  7  7 11  9  9  8 11  4  6 11  7  5 11 11 11
##  [5689]  4 11  5  5  3  5  5  7  6  7  9  8  6  4  9  6 10  4  8  5  6 12  4  8
##  [5713]  3  8  7  5 11  4  8  7  2  9  4  6  6 11  5  8  6  4  7  7  8 10  5  6
##  [5737]  9  7  6  5  3 11  7  8  7  5  3  9  6  3  5  9  9  6  8 10  9  8  3  6
##  [5761]  5 10 12  6  5  2 10  6  7  8  9  7 10  4  4  4  6  7  8  7 10  9 11  7
##  [5785]  7  9  7  3  7 10  4  7 11  5  6  5 10  7  6  8 10 10  4  7  6  7  9  7
##  [5809]  6  6  8  5  8  5 11  9  8  4  8  5  2  9  4  2  9 11  9  5  6  8  3  5
##  [5833]  8  3  6  6  8  9  9  9  5  6  8  7  6  6  9  7  8 10  7  7  4  7  6  2
##  [5857]  7  7  9  7  5  5  9  8  7 11 11  2 11  8  8  5  6  9  5 12 10  6  6  5
##  [5881]  7  4 10  9  6  8  8  8  5  9  8  8  9  8  7 11  6  6  7 10  6  6  7  4
##  [5905]  7  6  9  9  8  4  8  5 10  6  3  8  7  7  7  7  5  4  6 11  7 11  9  4
##  [5929]  3  7  8  5  3  9  7  8  7  8 10  9  4  5  7  5  9  7  6  8  8  8  7  7
##  [5953] 10  5  9  7 12 11  8  6  6  6  9  7  7  9 11  5  7  9  4  5  9  8  7  5
##  [5977]  4  6  7 10 10  6  6  9  7  7  6  3  8  8 11  7  3  4  7  6  8  7  4  7
##  [6001]  4 10  5  8  6 11  3  7 10  6  8  7  8 10 12  9  7 11  7  8  7  3  8  9
##  [6025]  4  9  7 10 11  4 12  7  9  5  5  6  7  9  9  7  4  6  7  6  7  9  8  9
```

```
##  [6049] 10  9  6  7  7  6  5  8  9  8 11  5  6 11  8 10  9  7  9  8  9  7  7  6
##  [6073]  6  5  6  6  6  8  7  5  8  4  9  3  9  2 10 10  7  9  7  8 10 10  6  6
##  [6097]  5 12 11  9  9  8  9  7  8  7  6  6 12  9 10 10  9  7  8  8  7  6  3  7
##  [6121]  8  8  4  6  8  8  9  5  6 12  3  5  3  8  9 12  5 10  5  6 11 11 11  2
##  [6145]  8  8  7  8  5  5  7  7  6  7  6  4  7 12  6  4  5  7  7  9  4  6 11 12
##  [6169] 10  6  9  4  4  8  5  8  6  9 10  5  7  8  6  5  7  8  2  8  7  4 10 12
##  [6193]  8 10  6  7 10 10  8  8 10  7  6  8 10  3  8  6  6  9  3  6 10  9  9  5
##  [6217]  9  7  7 11 11  8  8  8 12  6  8 10  9  8  4 10  6  9  9 11  6  6  4 11
##  [6241] 10 12 11  7 12  4  6 12  5  8  9 12 11  4  3  8  3  5  9  3  5  8 10  5
##  [6265]  7  2  7  7  9  6  3  8 10  6 10  6  3  7  6  8  7  4  7  8  7  7 10  6
##  [6289]  5  5 10  8  5  8  8  8  7  9  2  3  4  7  4  8  7  9  8  6  9 11  4  4
##  [6313]  5  7  3  7  8 11  8  8  6  5  7  8  4 10  6 12  7  7  3  6  7 11  6  3
##  [6337]  8  7  8  8  9  9 10  9  4 11  7  7  9  7 11  7  8  8  6  3  4  8  9  8
##  [6361]  7  5 11 10  6  2  7  2  7 12  6  6  8  3  8 10 11 10 12  9 10  9  9 12
##  [6385]  3  6  6  5  9  6  7  5  6  2  8  5  5  7  8  8 10  2  5  3  8  6  5  9
##  [6409]  8  6  9  7  3  6  7  7  8  6  5  5 10  5 12  7 10 12  8 10  5  6  8  8
##  [6433]  9  8  2 11  9  9  8  3  6  7  4  7  6  7  8  6  5  3  6  8  7  8 11  4
##  [6457]  8  8  6  3  9  3 10 10  4  4  7  6  4  9  5  9  7 10  3  9 10  3  4 10
##  [6481] 10  6  2  9  5 11  4  3  7  8 10  8 10  2  7  9 10  7  7  7  5  9  3  6
##  [6505]  4  8  9 10  6  5 10 10  5  5  7  2 11  5  7  4  8  5  7  7  8  8  8  5
##  [6529]  6  7  7  5  5  5 11  4  3  5  5  6  3  8  6  5  9  8  4  8  7 11  7  5
##  [6553]  6 10 10  7  5  5 10  7 10  7  6  8  4  7 10  7  4  7  2  5  8  9 10  6
##  [6577]  5  7  7  8  9  5  7 12  6  4  7  7  4  5  9  7  9  8  9  9 10  8  6  4
##  [6601]  6 11  9 10  8  8 12  8  6 10  7  8  8 10  4  5  9  7  8  7  5  6  7  6
##  [6625]  5 10  9 11  9  6  8  5  7  7  2  5  5  5  5  7  4  8 10  6  6  6  7  7
##  [6649] 10  6 10  9  7  8  6  5 10  6 10  8  4  5  9  7  6  8  5  7  2  8  4  8
##  [6673]  7  5 12  6  8  6  4 11  9  5 10  7  7  7  6  5  5  9  8  5  7  7  4  8
##  [6697]  6  5  7  7  3  8  7  5  4  5  6  9 11  7 10  8  6  9  7 10  9 10  7  7
##  [6721]  7  8  7  4 10  6  8 11 11  4 11  9  6  7  6  7  7 11  6  6 11  7  3  6
##  [6745]  6 11  7  7  9 10  2  9  9  9 12  4  4  6  3  7  7 10 10  6  7  4  7  5
##  [6769]  4  7  6 10 10 10  8  5  2  6  6  4  8  7  6  9  8  9  7  6 12  9  2  4
##  [6793]  4  8  9  9  8 10  7  9  4  8  6  9  6  4  8  9 11  6  4  8  3  6  7 11
##  [6817]  8  6  5 11  6  6  9  4  8  7  5  8  9  9  8  5 12 10  3  4  3 10  8  8
##  [6841]  5  4  5  2  7  5  9  6  6  4  8  4  3  8  7  9  6 11  5  5  8 10  8  8
##  [6865] 10  6  3  3  7  7  6  5  4  7  9 10  9  9 10  6 10  3  9 11  7  7  7 11
##  [6889]  4  6  9  6  7  8  7  7  8  5  7  7  3 12  5  6 12  5  7  8 10  5  3  2
##  [6913]  9  6  7 10  6  4  8  8  5  9  9  7  6  6 11  6  6 11  7  2  7  3  8  7
##  [6937]  5  9  5 10  5  7  8  7  9  9  8  4  9 10 12  8 10  5  7 10 11  9  8  6
##  [6961]  8  9  2  8  4 11  9  4 11  4 10 10  8  8 11  7  9  4  6  7  3  9  8 12
##  [6985]  5  9  7  3  5  2  7 10  6  9  6  7  3  9  8  8  5  6  7  6  9  2  8  7
##  [7009]  6  7  6  8  6  5  9  6  6  9  8  5 11  6 11  5 10  8  9  7  5 10  8  7
##  [7033]  2 11  5  9  6  5  5  8  8  4  5  6  4  7 11  4  5  5  5  6  7  8  5 11
##  [7057]  7  9 10  3  7 12 12  7  6  7  6  3  4 12  7  5  4  8  4  7  4  8  8  8
##  [7081]  8  4  3  9  7  6  3  5  7  7  4 10  6  8  5  8 10  3  8 10  4  7  6  8
##  [7105]  6  5 12  5  6  9  7 10  6  6  8  6 11  3 10  5  8  8  4  9  3  8 10  5
##  [7129]  4  6  3 11  9  8  7  5  5  7  7  9  3  7  7  7  3  5  9  8  6 12  7  4
##  [7153]  3  6  6  6  5  7  8  5  6  7 10  5 11  8  7  6  4  9  4  8  4  5  7 11
##  [7177]  9  7  6  8 10  9 12  9  6  7  3  6  7 10  2  6  5  8  3  7  7  6  4 11
##  [7201]  8  7 11  7  8 11  7  7  8 10  4  5  6  8  4  3  6  8  8  5  9  7  5  6
##  [7225]  3  8  3 10  7  9  4  6  4  8  8  9  7  8  9  5  9  6  6 11  4  7  8  8
##  [7249]  6 10  6 10 10  2  3 12 11 10 11  6 10  7  6  6  4  8  8  5  9 11 10  9
##  [7273]  9  8  8  9  7 10  3  9  3 12  5  6 10  6  5  8  8  8  6  7  7  5  8  5
##  [7297]  6 11 10 10  5  6  5  7  7  7  7  6  9  5  7  2  6  3  7  9  8  5  9  8
##  [7321] 11  7  8  6  7  7 11 12 10  3  7  5  5  7  8  6  9  7  6  7  7 12 10  9
```

```
##  [7345]  8  4 10  7  7  7  7  9 10  4  9  7 11 11  6  8  9  7  3  7  4  3 10  5
##  [7369]  5  2  8  8 10 10  6  7  6 10 11 11  7  9  6  6  3  7  5  6  2  6  8  9
##  [7393]  9  5  7 11  7  3  2 10  6  8  8  6  8  9  8  7  6  2  6  5  9  6  7  9
##  [7417]  6  8 10 12  7  8  4  9  7  7  7  9  9  8  2  8  9 10  6  3  7  6 11  8
##  [7441]  7 12  7  6  5  7  8  6  6  8  9  4  8  5  8  8 10  4  7  5  8  8  3  4
##  [7465] 11  5  8  9  4  5  6  4  6  7  6 11  8 10  5  8 10  5  8  6  7  9  7 10
##  [7489]  5 10  6 10  6  8 10 10  4  7  7  8  8 10  7  6  4  5  6 10  7  5  7  2
##  [7513]  6 10  7  8  8  7 12  5  8  6  6  7  7 10  8  4  8 10  4  7  7  8 10  6
##  [7537]  9  5  8  6  2  6  8  2  9  8  3 10  4  4  6  7  7  9  7 12  8  5  6  8
##  [7561]  2  9  5  5  4  7 12  4  5  5  6 10  7  4  7 10  5  4  9 10 10  4  5  8
##  [7585]  7  6  9  8  2 10  5  5 10  9 11  5 10 10  6  8  3  5  4 11  7  5  6 10
##  [7609]  2  7  5  4  8  7 10  8  9 10  7  5  7  8  5  8  8  7  7  6  5  5  7  9
##  [7633]  4  7  2  6  7  9  5  7  5  8  7 11  6  5  9  5  7  2  9 10  4  6  8 10
##  [7657]  9  5  5  6 10 10  3  7  6  5  7  5  8  8  8  4  7  8  5  5  7 10  6  3
##  [7681]  4  8  3  8  4  8  6  8  5  3  5  2  4  8  7  6  9  6  7  9  7  8 10  9
##  [7705] 11  6  7  5  7  4  7  3  5  5 12  7  7  5 10  6  5  3 11  3  8  6 10  8
##  [7729]  9  7  7  4  9  6 10  7  6  4  4  5  3 11 11  7 12  8  4  9 10  7  8  8
##  [7753]  8 11  8  6  8  5 10  3  9  9  6  7  9  6  9  9  5 10  5  9  9 10  7  8
##  [7777]  9  9  3  9  9  8 10  4  9  9  8  2 11 10  8  6  7  7  8  9  9  5  6  4
##  [7801]  6  8  5 10  9  5 11  4  6  3  8  9  5  8  6  8 11 10  7  9  9  8 12  5
##  [7825]  7  7  6  7 10  6  4  8  2  2  5  8 10  7  4  4  7  4  4  7  4  2  4  8
##  [7849]  8  8  2 10  9  6  5  5  9 10  4  8  5 10 12  8  8  8  9  7 10  9  8  5
##  [7873]  4  7  4  5  3  9  3  6  9  9  5  9  9  4  5  6  8  7 10  5  6  3  6  8
##  [7897]  9  7  8  6  8  9  3  2  9 12  8  3  5  8  8  7 10 10  6  7  5 11  8  6
##  [7921] 11  7  6  3  6  9 10  3  8  8  8 11 11  7  5  4  5  9  7  7  3  7  2  6
##  [7945]  3  3  6  9  8  8  7  8  4  3  4  7  9  7  8  4 11  9  7  7  7  7  6  8
##  [7969]  2  5  7  4  9  6  5  6  9 11  7  8  9  8 11  6  4  3  6  6  9 10  7  6
##  [7993]  4  4 10 11  7  9  9  7  9  7 10  7  3  8  7  4  9  7 10  7  7  7  8  6
##  [8017]  6  9  9  8  5  5  9  6  8  8  3  4  8  6  8 11  9  6 11  9  7  4  5  6
##  [8041]  2  5  7  9  6  9  5 11  7  9  9  8  5 11 10  8  6  5  6 10  6  2  5  9
##  [8065]  6  9  5  3  8  6  6  6  7 10  2  7  6 10 10  2  7  7  4 11  6  3  8  9
##  [8089]  9  3  9  3  5  9  6  7  2  6  5  5  5 11  5  6  7  3 10 10  9  9  5  5
##  [8113]  7 11  6  3  6  7 10 10 11  9 10  4  4  9  5  2  9  4  6  3  9 11 10 10
##  [8137]  7  7  7  3  9  9  6  6  2  9  3  7  8  7 10  9  4  8  7  7  4  7  5  5
##  [8161]  8  5  3  4  7 10  5  8  7  8  7  5  2 10  5  2  3  5  6 11  3  3  4 10
##  [8185]  5  6  5  6  9  9  9  7  5  7  8  6  4  6  3  7  5 10  9  9  9  9  5  3
##  [8209]  9  3  6  7  6  8  7  8  7  7  4  6  8  5  5 12 10  8  7  2 11  7  6 10
##  [8233] 12  5  9  4  5  3  4  9  6  9  7 10 10  7  6  7  5  9  6  8  5  3  9  8
##  [8257]  5  5  5  5 11  2 10  7 11  9  6 11  7  7  5  5 11 11  8  6  4  4  8  8
##  [8281]  4  4  8 10  5  4  3 10  6  7 11  8  7  3 10  6 12  8  5 10  9  5  7 11
##  [8305] 11  8  2  8 11 11  2  4  5  8  8  7  8 12  7  9  6  9  5  8  9  7  8  7
##  [8329]  9  6 10  7  4  7  3  6  7 10  7  7 11 12  4 10  6  9 10  8  7  5 12  7
##  [8353]  5  4  7  8  8  2  6  8  8  9 10  7  8  8  3  5  8 11  5  7  4  5 11  7
##  [8377]  7  6  2  5  8  7  8  7  2 10  4  9  7  4  5  8 10 11  7  8  6  6  7 11
##  [8401]  4  9  9  5 12  2  3  8 10  5  8  9  9  9  6  7  7  3 10  5  5  7  6  7
##  [8425]  8 12  7  2 10  3 10  7  5  9  8  5  5  2  9  8  5 11  7  7  5  7  5  6
##  [8449]  5  6  4 11  8  6  4 10  9  7  5  4 11  5  5  4 10  4 10  7  9  6  8  9
##  [8473]  9  4  5 10  5 10  5  7 11  2 10  7  4  8  8  6  4  5  2  3  8 11 10  6
##  [8497]  9  4  8  9 11  5  7  9  4  8  6  2  9  6  9 12  8  6  4  7  5  8  7 10
##  [8521]  2 10  9 10  7 10  4  9  8  8  7  7  7 11  5  9 10  7  3  8 10  7  3  7
##  [8545]  2  6  3 11  8  8  8  8  3  3 10  2  7  6  8  5 12  7  8  4  4  6 12  4
##  [8569]  6  9  8  7  8  6  6 11  6 10  5  8  9  3  6  8  5 10  7  6  7  6 10  3
##  [8593] 10  8  7  9  9  5  5  3  3 10  9  7  8  9 10  3  5  7 11  7  7  8  9  6
##  [8617] 10  7  5  5  9  5  7  5  3  9  5  7 12  5  5  9  9  7 10 10 10  4  5  3
```

```
##  [8641]  7  9  6  6  9  6  7  5  5  5  7  5  6  5  7  9  8  5  7  5  6 10 10  6
##  [8665]  7  7  2  7  8  7  2  9  8  7  7  9  2  7 11  6  8  7  5 11 11  7  3  4
##  [8689]  6  8  7  7  4  8 11  3  6 10  7 11  7 11 10  9  7 10  6  5  7  2  5  7
##  [8713] 11 10  8 12  8  5  6 11  8  8  5  6  6  2  7  4  3  3  9  3 10  9  5 12
##  [8737]  5 11  9  8  6  8 11  6  3  4  5  8  9  8  8  9  4  5  9  4  7  6  8  5
##  [8761]  5  5  8  5  3 10  3  7  7  7  7  2  4  6  8  3  4  9  7  5  4  8  7  7
##  [8785]  7  6  6  9  5  4  7  4  8  8  9  8  5  7 11  8 11  9  3  5  7  3 10 10
##  [8809]  8 10  3  9  5  7  5  5 10 10  9  8  4  6 10 10  2  6  7  5  8  7 11  5
##  [8833]  5  8  6  9  5  8  5  5  7  3  8  7  7  5  7  4  6  5  5 11  8  9 12  8
##  [8857]  7  6 11  5  8  4  4  5  3 10  3  9  2  4 10  9  8  5  4  9  8 10  9  7
##  [8881]  9 10  6 12  9 10  4  8  9  8 10  7  6  8 12  6  4 10  8  6  8 10  8 12
##  [8905]  9  6  6  8  5  6  5  6  8  4  7  9  6  9  4  7  7  5 10  7  7  7  5 10
##  [8929] 10  7  2  8  9  8 11  5  7  8  9  7  8  4  6  9  9  7  7  4  5 10  8  7
##  [8953]  7  7  5  4  8  6 10  3  6  8  7  5  6 11  4  2  8 10  6  7  6  8  7  2
##  [8977]  6  5 10  4 10  3  7  6 10  6  4  4 10  8  8  5  5  8  9  5  9  6  9  5
##  [9001]  5  4 10  4 10  8  8  8  9  6  7  4  7 10 12  6 10  9  8  6 11  4  8  6
##  [9025]  7  7  5 10  5 11  8  4  7 10 11  5 11  7  8  6  2  8 10  6  5  4  7  7
##  [9049]  6 11  7  6 12  6  5 10  4  6  5  4  6  7  9  8  8  9  5  8  4  6  9  9
##  [9073]  6  8  6  7  7  3  8  9  7  9 11  5  3 10  9  7 12  5  5  9  8  6  9  4
##  [9097]  2  8 10  8  5 10 10  7 10  7  3  8  9  5  4 11  8  7  4  7  8 11  7  8
##  [9121]  6 10  8 11  5  8  9  4 11 10  7 10  7  5 10  7 12 10 11 10 11 10  7 10
##  [9145]  4  2 11  9  9  8  6  6  7  4  5  6  9  4 10  9  9  6  8 12  7 11  9  9
##  [9169]  9  4 11 12  2  8  9  3  6  4  2  5  7  4  9 10  8  9  9  8 10  9  7 10
##  [9193]  5  6  8  7  7  9  4  7  3 10  9  8  5  4  7  7  5 11  6  6 11  8 10  9
##  [9217]  9  7  3  8  8  5  4  5  7 12  5 11  8  3  8  7  4  4  5 12  7  5  6  8
##  [9241]  8  8  2 11  8  6  4  3 10  7  5 11 11  7  4  6  7  6  3  7 10  8  7 11
##  [9265]  7  9  5 12  8  5  7  7  8 10  9  9  7  9  4  6  9  7  9  9 11  7  7  4
##  [9289]  4  5 10  8  7  4 11  5  6 10 10 10 10  5  2 11  2  7  2  3  7  6 11  7
##  [9313] 10  2  5  8  8  3 12  7  3  3  7  6  5  9 11  8  8  9  6  6  5  7  6  7
##  [9337]  5  4  8  6  7  8  7  7  9  4  8  6 10  5  7  6  6 11  4  5  9  7  7  6
##  [9361] 10  7  8  4  6  5  6  7  6  5  3  6 10 11  2 12  2  8  6  8 10  8  3  3
##  [9385]  7  7  7  6  7  7  5  9  9  7  9 11  7 12  5  6 11 11  7  9  5  8  5  3
##  [9409]  4  8  4  8 10  7  9  9  5  6  5  7  2  4  7  7  7 10  6  3 11  5 10  3
##  [9433] 10  5 10  7  8  5 12  7 10 12  5  9  7  4 10  7  9  2  7  9  9  5  9  3
##  [9457]  3  4  9 11  9 10  4  8 12  6  6  4  5  4  6  7  8 11  4 10 10 10  4  6
##  [9481]  6 10  8  6 12 10  9  8  5 12  7 11  9  6  5  9 10  5  9  5  8  7 11  9
##  [9505]  7  9  5 11  8  5  9  3 10 10  5  8  8  5  3  8  7  5  7  7  5  6  7 11
##  [9529]  5  3  7  5  7  3  8  9  2  9  7  8 10  7  7  5 12  4  8  4  6  8 10  7
##  [9553]  8  6  7  7  7  9 10  9  7  5  7  8  5 11  7  8  6  5  6  9  5  6  6  6
##  [9577]  9  8  6  9  7 10 11  5  8 10  7  8 10 11  4  7 10  7  7  4  9  7 11  7
##  [9601]  7  7  5  4  9  8  5  7  8  4  6  2 10  8  8  8  7 12  6 10  3  5  8  5
##  [9625]  4  5 10  2  8 11  5  9  7  7  7  3  8 10 12  8  9 10 11  9  9  6  3  7
##  [9649]  7  3  5  3  9  7 11 11 12  6  5 12  5  4  6  6  8 11  4  8  7  2 11  9
##  [9673]  5  7  9 10  6  8  7  4  3  8  7  4  9  8  6  6  8  6  8  7  8 10  7  2
##  [9697]  6  3  4 12  9  6  6  3  8  5  8 10 12  7  6  6 10  5 11  6  7  7  9  4
##  [9721]  5  6  7  3  7 12  6  7  8  8  5  2  8 10  5  5  5  9  5  8  5  3  4  4
##  [9745]  5  3 10 10  8  9  8  9  6  2 10  3 11 10  8  9  4  2  7  6  6  9  9  8
##  [9769]  6  5  6 11  9  5  4  6  8  7  8  7  9  5  6  6  9  6 10  8  5  7  2  9
##  [9793]  4  6  2  6 10 12  8  8  9  5  8  6  2  9  5  6  9  5  5  5  6  4  7  8
##  [9817]  8  8  3  5 12  6  5  8  9  5  8  5 11  6  5 12  9  7  8  8  7  5 11  6
##  [9841]  7  5  5  7  7  8  7  3  7  3  8  3  4 11  5  6  8  5  3 11  9  7  5  6
##  [9865] 11 10  7  6 10 12  8  3 11  3  9  7  5  7  3  8 12  4  8  9  9  9  6  7
##  [9889]  5  6  4  8  3  5  6  8  6  8 12 10  5 11 10  7  9  9  7  7  9  9  5  6
##  [9913]  3  6  7  6  9  7  4  5  7  8  9  7  8  4  9  7  4  2  8  4 11  7 11  7
```

```
##  [9937]  5  3 10  6  6  6  5  7 10 10  6  5  7 10  6  8  8  3 10  3  4  6  6  8
##  [9961]  9  4  6  7  5  8  9 10  7  8  8 11  3  8  2 10  7  3 10 10  7  4  9  8
##  [9985]  8  8  4  7  3  9 12  5  4 10  8 12  6  5  8 11
```

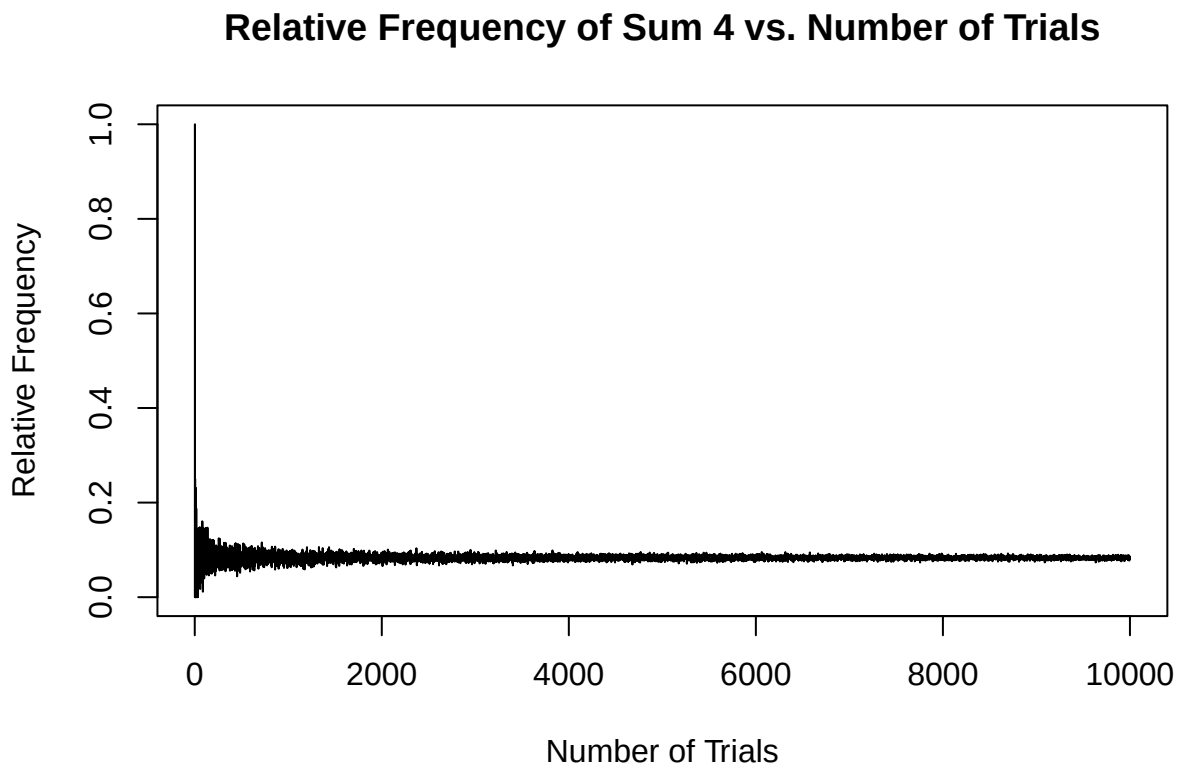
Use "sum_die" to calculate the empirical probability of getting the sums 2, 3, 4, ..., 12. Compare your answers with the theoretical values.

```
X <- 10000
sum_die <- replicate(X,roll_dice)
sum_sample <- sample(c(1,2,3,4,5,6),size = X, replace = TRUE)
sum(sum_sample[1]+sum_sample[2])
```

```
## [1] 7
```

Plot the following: relative frequency of getting sum equal to 4 as a function of number of trials, for this experiment.

```
num_trials <- 10000
relative_frequencies <- numeric(num_trials)
for (i in 1:num_trials) {
  outcomes <- replicate(i, roll_dice())
  relative_frequencies[i] <- sum(outcomes == 4) / i
}
plot(1:num_trials, relative_frequencies, type = "l", xlab = "Number of Trials", ylab = "Relative Freque
```



Coin Tosses for Binomial Distribution

Suppose we toss a coin with $P(H) = 0.8$, n times independently. Let X be the the random variable that counts the number of heads in n tosses.

n = 2

Setup a function that simulates the random variable X when we toss the coin two times.

```
toss <- function(){
  coin_toss <- sample(c("H", "T"), size = 2, replace = TRUE, prob = c(1/2, 1/2))
  return(coin_toss)
}

toss()
```

```
## [1] "H" "T"
```

Use the above function to calculate the emprical probability distribution table for X. Compare this with the theoretical probability distribution of X.

```
num_trials <- 10000

# Initialize a table to store the counts of each outcome
outcome_table <- table(replicate(num_trials, paste(toss(), collapse = "")))

# Calculate empirical probabilities
empirical_probs <- outcome_table / sum(outcome_table)

# Create a data frame for the probability distribution table
probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1      HH               HH           0.2599
## 2      HT               HT           0.2447
## 3      TH               TH           0.2499
## 4      TT               TT           0.2455
```

Calculate emprical estimates for the expected value and the variance of X. Compre these with the actual theoretical calculations.

```
num_trials <- 10000

sum_outcomes <- 0
sum_squares <- 0

# Simulate coin tosses and calculate the sum and sum of squares
for (i in 1:num_trials) {
  result <- sum(as.integer(toss() == "H"))  # Count "H" as 1, "T" as 0
```

```
  sum_outcomes <- sum_outcomes + result
  sum_squares <- sum_squares + result^2
}

# Calculate empirical estimates for the expected value and variance
mean_estimate <- sum_outcomes / num_trials
variance_estimate <- (sum_squares / num_trials) - (mean_estimate^2)

mean_estimate
```

```
## [1] 1.0129
```

```
variance_estimate
```

```
## [1] 0.5019336
```

$\mathbf{n = 3}$

Setup a function that simulates the random variable X when we toss the coin three times.

```
toss <- function(){
  coin_toss <- sample(c("H", "T"), size = 3, replace = TRUE, prob = c(1/2, 1/2))
  return(coin_toss)
}

toss()
```

```
## [1] "H" "H" "H"
```

Use the above function to calculate the emprical probability distribution table for X. Compare this with the theoretical probability distribution of X.

```
num_trials <- 10000

# Initialize a table to store the counts of each outcome
outcome_table <- table(replicate(num_trials, paste(toss(), collapse = "")))

# Calculate empirical probabilities
empirical_probs <- outcome_table / sum(outcome_table)

# Create a data frame for the probability distribution table
probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1     HHH              HHH           0.1215
## 2     HHT              HHT           0.1241
## 3     HTH              HTH           0.1241
## 4     HTT              HTT           0.1317
```

```
## 5     THH             THH           0.1266
## 6     THT             THT           0.1282
## 7     TTH             TTH           0.1203
## 8     TTT             TTT           0.1235
```

Calculate emprical estimates for the expected value and the variance of X. Compre these with the actual theoretical calculations.

```
num_trials <- 10000

sum_outcomes <- 0
sum_squares <- 0

# Simulate coin tosses and calculate the sum and sum of squares
for (i in 1:num_trials) {
  result <- sum(as.integer(toss() == "H"))  # Count "H" as 1, "T" as 0
  sum_outcomes <- sum_outcomes + result
  sum_squares <- sum_squares + result^2
}

# Calculate empirical estimates for the expected value and variance
mean_estimate <- sum_outcomes / num_trials
variance_estimate <- (sum_squares / num_trials) - (mean_estimate^2)

mean_estimate
```

```
## [1] 1.5012
```

```
variance_estimate
```

```
## [1] 0.7625986
```

$\mathbf{n = 4}$

Setup a function that simulates the random variable X when we toss the coin four times.

```
toss <- function(){
  coin_toss <- sample(c("H", "T"), size = 4, replace = TRUE, prob = c(1/2, 1/2))
  return(coin_toss)
}

toss()
```

```
## [1] "T" "H" "T" "T"
```

Use the above function to calculate the emprical probability distribution table for X. Compare this with the theoretical probability distribution of X.

```r
num_trials <- 10000

# Initialize a table to store the counts of each outcome
outcome_table <- table(replicate(num_trials, paste(toss(), collapse = "")))

# Calculate empirical probabilities
empirical_probs <- outcome_table / sum(outcome_table)

# Create a data frame for the probability distribution table
probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

probability_table
```

```
##    Outcome Probability.Var1 Probability.Freq
## 1     HHHH             HHHH           0.0652
## 2     HHHT             HHHT           0.0609
## 3     HHTH             HHTH           0.0628
## 4     HHTT             HHTT           0.0632
## 5     HTHH             HTHH           0.0620
## 6     HTHT             HTHT           0.0661
## 7     HTTH             HTTH           0.0645
## 8     HTTT             HTTT           0.0613
## 9     THHH             THHH           0.0614
## 10    THHT             THHT           0.0606
## 11    THTH             THTH           0.0609
## 12    THTT             THTT           0.0618
## 13    TTHH             TTHH           0.0607
## 14    TTHT             TTHT           0.0622
## 15    TTTH             TTTH           0.0645
## 16    TTTT             TTTT           0.0619
```

Calculate emprical estimates for the expected value and the variance of X. Compre these with the actual theoretical calculations.

```r
num_trials <- 10000

sum_outcomes <- 0
sum_squares <- 0

# Simulate coin tosses and calculate the sum and sum of squares
for (i in 1:num_trials) {
  result <- sum(as.integer(toss() == "H"))  # Count "H" as 1, "T" as 0
  sum_outcomes <- sum_outcomes + result
  sum_squares <- sum_squares + result^2
}

# Calculate empirical estimates for the expected value and variance
mean_estimate <- sum_outcomes / num_trials
variance_estimate <- (sum_squares / num_trials) - (mean_estimate^2)

mean_estimate
```

```
## [1] 2.0124
```

```
variance_estimate
```

```
## [1] 0.9958462
```

Coin Tosses for Negative Binomial Distribution.

Suppose we toss a coin with $P(H) = 0.8$ until we get r heads.

r=1

Setup an experiment that counts the number of tails until the first head. Run 10000 simulations of this experiment, and use these to find estimate for the expected number of tails until the first head. Match this value with the theoretical/calculated value.

```
count_tails_until_head <- function() {
  tails_count <- 0
  while (TRUE) {
    # Simulate a coin flip (0 for tails, 1 for heads)
    coin_flip <- sample(0:1, 1)
    tails_count <- tails_count + 1
    if (coin_flip == 1) {  # Head
      break
    }
  }
  return(tails_count)
}
num_simulations <- 10000

# Vector to store the results
results <- numeric(num_simulations)

# Run the simulations
for (i in 1:num_simulations) {
  results[i] <- count_tails_until_head()
}
expected_val <- mean(results)
expected_val
```

```
## [1] 1.9985
```

r = 2

Setup an experiment that counts the number of tails until the second head. Run 10000 simulations of this experiment, and use these to find an estimate for the expected number of tails until the second head. Match this value with the theoretical/calculated value.

```
count_tails_until_second_head <- function() {
  count_tails <- 0
  head_count <- 0
```

```
  while (head_count < 2) {
    # Simulate a coin flip (0 for tails, 1 for heads)
    flip <- sample(0:1, 1)

    if (flip == 1) {
      head_count <- head_count + 1
    }

    count_tails <- count_tails + 1
  }

  return(count_tails)
}
num_simulations <- 10000

# Vector to store the results
results <- numeric(num_simulations)

# Run the simulations
for (i in 1:num_simulations) {
  results[i] <- count_tails_until_second_head()
}
expected_val <- mean(results)
expected_val
```

```
## [1] 3.9875
```

r = 3

Setup an experiment that counts the number of tails until the third head. Run 10000 simulations of this experiment, and use these to find an estimate for the expected number of tails until the third head. Match this value with the theoretical/calculated value.

```
count_tails_until_second_head <- function() {
  count_tails <- 0
  head_count <- 0

  while (head_count < 3) {
    # Simulate a coin flip (0 for tails, 1 for heads)
    flip <- sample(0:1, 1)

    if (flip == 1) {
      head_count <- head_count + 1
    }

    count_tails <- count_tails + 1
  }

  return(count_tails)
}
num_simulations <- 10000
```

```r
# Vector to store the results
results <- numeric(num_simulations)

# Run the simulations
for (i in 1:num_simulations) {
  results[i] <- count_tails_until_second_head()
}
expected_val <- mean(results)
expected_val
```

```
## [1] 5.9798
```

Simulating the Monty-Hall Problem

An interesting application of conditional probability is the Monty-Hall problem, whose solution intrigued many, including career mathematicians (life is hard!). There is a lot of trivia around this problem in the internet, a good place to start would be the Wikipedia article here: https://en.wikipedia.org/wiki/Monty_Hall_problem

In this project we will understand the solution of this problem using simulations.

Monty-Hall Three doors

Run a simulation to check that the probablility of winning increases to 2/3 if we switch doors at step two in the regular 3-door Monty-Hall problem.

Set up the experiment two functions "monty_3doors_noswitch" and "monty_3doors_switch" (these functions will have no input values):

```r
monty_3doors_noswitch <- function(){
    prize <- sample(1:3, 1)
    choice = 1
    if (prize == choice){
      1
    } else{
      0
    }


}

monty_3doors_switch <- function(){

  prize <- sample(1:3,1)
  choice = 1

  if(prize == choice){
    notprize = sample(2:3,1)
    if(notprize == 2){
      choice = 3
    } else{
      choice = 2
```

```
    }
  } else if (prize == 2){
    notprize = 3
    choice = 2
  } else{
    notprize = 2
    choice = 3
  }


  if(choice == prize){
    1
  } else {
    0
  }




}

monty_3doors_switch()
```

```
## [1] 1
```

Use your two functions and the replicate function to compute the empirical probablility of winning for the two experiments. Compare your answers with the actual theoretical predictions.

```
num_trials <- 10000

# Initialize a table to store the counts of each outcome
outcome_table <- table(replicate(num_trials, paste(monty_3doors_switch(), collapse = "")))

# Calculate empirical probabilities
empirical_probs <- outcome_table / sum(outcome_table)

# Create a data frame for the probability distribution table
probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.3396
## 2       1                1           0.6604
```

```
num_trials <- 10000

# Initialize a table to store the counts of each outcome
outcome_table <- table(replicate(num_trials, paste(monty_3doors_noswitch(), collapse = "")))

# Calculate empirical probabilities
```

```r
empirical_probs <- outcome_table / sum(outcome_table)

# Create a data frame for the probability distribution table
probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.6717
## 2       1                1           0.3283
```

```r
print("The first table is the probabiltiy associated with you switching which results in a win frequency
      the second table is the probability associared with you now switching which results in a win frequ
      This reflects the actual theoretical probabilites.")
```

```
## [1] "The first table is the probabiltiy associated with you switching which results in a win frequenc
```

Monty-Hall with Ten doors.

Repeat the Monty Hall experiment now with 10 doors. Recall the game is as follows: Step 1: you choose one door at random. Step 2: Monty opens 8 (out of 9 doors) that do not have the prize. Step 3: you either switch or don't switch.

Set up the experiment two functions "monty_10doors_noswitch" and "monty_10doors_switch" (these functions will have no input values):

```r
monty_10doors_noswitch <- function(){
   prize <- sample(c(1:10), 1)
    choice = 1
    if (prize == choice){
      1
    } else{
      0
    }
}

monty_10doors_switch <- function(){
  prize <-sample(c(1:10), 1)
  choice = 1

  if(prize == choice){
    close<- sample(c(2:10),1)
    choice = close
  } else {
    choice = prize
  }

  if(choice == prize){
    1
  } else {
    0
  }
```

```
}
```

Use your two functions and the replicate function to compute the empirical probablility of winning for the two experiments. Compare your answers with the actual theoretical predictions (you need to calculate these).

```
  num_trials <- 10000

  # Initialize a table to store the counts of each outcome
  outcome_table <- table(replicate(num_trials, paste(monty_10doors_switch(), collapse = "")))

  # Calculate empirical probabilities
  empirical_probs <- outcome_table / sum(outcome_table)

  # Create a data frame for the probability distribution table
  probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

  probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.0969
## 2       1                1           0.9031
```

```
  num_trials <- 10000

  # Initialize a table to store the counts of each outcome
  outcome_table <- table(replicate(num_trials, paste(monty_10doors_noswitch(), collapse = "")))

  # Calculate empirical probabilities
  empirical_probs <- outcome_table / sum(outcome_table)

  # Create a data frame for the probability distribution table
  probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

  probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.9012
## 2       1                1           0.0988
```

```
  print("The first table represents the probability of switching after 8 doors are opened. This results
        the second table represents the probability of not switching after 8 doors are opened. This resu
        The accurately aligns with the actual probabilities. For not switching you have to choose the pr
        is a 1/10 chance. However, if you picked a door without a prize intially (9/10) then switched y
```

```
## [1] "The first table represents the probability of switching after 8 doors are opened. This results
```

Monty-Hall 10-doors (modified).

Consider the following modified Monty-Hall game with 10 doors. Step 1: you choose one door at random. Step 2: Monty opens 7 (out of 9 doors) that do not have the prize. Step 3: you either stick with your original choice, or choose between one of the two unopened doors.

Set up the experiment two functions "monty_10doors_mod_noswitch" and "monty_10doors_mod_switch" (these functions will have no input values):

```
monty_10doors_mod_noswitch <- function(){
     prize <- sample(c(1:10), 1)
     choice = 1

     if(prize == choice){
       1
     } else {
       0
     }


}
#my_door_and_prize_door <- c(my_door, prize_door)
#not_my_door_nor_prize_door <- doors[!doors %in% my_door_and_prize_door]

monty_10doors_mod_switch <- function(){
  prize <- sample(c(1:10),1)
  choice = 1

  if(prize == choice){
    #you can't win if you choose the prize initially
    0
  } else {

    if(prize == 2){
      door_choose <- sample(c(3:10),1)
    } else if(prize == 10){
      door_choose <-sample(c(2:9),1)
    } else {
      end = prize -1
      start = prize +1
      door_choose <-sample(c(2:end, start:10 ), 1)
    }

    choose <- sample(c(door_choose, prize), 1)

    if(prize == choose){
      1
    } else {
      0
    }


  }
```

```
}
```

Use your two functions and the replicate function to compute the empirical probablility of winning for the two experiments. Calculate the theoretical probability of winning a prize if you switch, and if you don't switch. Compare your values to the empirical probabilities.

```
  num_trials <- 10000

  # Initialize a table to store the counts of each outcome
  outcome_table <- table(replicate(num_trials, paste(monty_10doors_mod_switch(), collapse = "")))

  # Calculate empirical probabilities
  empirical_probs <- outcome_table / sum(outcome_table)

  # Create a data frame for the probability distribution table
  probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

  probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.5462
## 2       1                1           0.4538
```

```
  num_trials <- 10000

  # Initialize a table to store the counts of each outcome
  outcome_table <- table(replicate(num_trials, paste(monty_10doors_mod_noswitch(), collapse = "")))

  # Calculate empirical probabilities
  empirical_probs <- outcome_table / sum(outcome_table)

  # Create a data frame for the probability distribution table
  probability_table <- data.frame(Outcome = names(empirical_probs), Probability = empirical_probs)

  probability_table
```

```
##   Outcome Probability.Var1 Probability.Freq
## 1       0                0           0.8946
## 2       1                1           0.1054
```

```
    print("The first table represents the probability of switching after 7 doors are opened. This result
        the second table represents the probability of not switching after 7 doors are opened. This resu
        The accurately aligns with the actual probabilities. For not switching you have to choose the pr
        is a 1/10 chance. However, if you picked a door without a prize intially (8/10) then switched yo
        to the prize. So (8/10) * (1/2) = 0.4 ")
```

```
## [1] "The first table represents the probability of switching after 7 doors are opened. This results i
```

Not for submission: Play with this modified setup, for example Monty opens k doors at step 2 etc with k any number between 1 and 8.

(Bonus) Monty Hall with n-doors.

This is for your curiosity, you don't have to turn this in. However, if you work this general case out, you can use it to solve the earlier versions of the problem.

Repeat the Monty Hall experiment now with n doors. Recall the game is as follows: Step 1: you choose one door at random. Step 2: Monty opens n-2 (out of n-1 doors) that do not have the prize. Step 3: you either switch or don't switch.

Set up the experiment two functions "monty_ndoors_noswitch" and "monty_ndoors_switch" (these functions will have input value as n):

```
monty_ndoors_noswitch <- function(n){

}

monty_ndoors_switch <- function(n){

}
```

Use your two functions and the replicate function to compute the empirical probablility of winning for the two experiments. Compare your answers with the actual theoretical predictions.

Bayes's Theorem and Witnesses

Recall the hit-and-run example from the asynchronous lecture and HW4. We noted that the Bayes' probabilities are calculated as the following formula

$$P(B|W) = \frac{pq}{pq + (1-p)(1-q)},$$

where $P(B) = q$ and $P(W|B) = p$ (check notes). We also noted in HW4 that if $P(W|B) = p > \frac{1}{2}$ we will have

$$P(B|W) > P(B).$$

In this problem, we will explore how changing the level of Witness-Reliability, that is p, will affect the chance of getting a Blue Cab driver arrested.

Recall that $P(B)$ is the probability that the Blue cab is at fault with no added information. We will now run multiple iteration of Bayes' rule, and at every iteration, we will update the value of $P(B)$ with the value of $P(B|W)$ from earlier iteration. From our experience from the homework, if $p > \frac{1}{2}$, we must have $P(B) \longrightarrow 1$ as the number of iterations increase (intuitively, the probability that the Blue cab is at fault will go to 1 as the number of "reliable witness" who say that "the Blue Cab is at fault" increases).

Write a function called "bayes" that takes input q, p and outputs $P(B|W)$ (use the formula above)

```
bayes <- function(q, p){
  return(p*q/(p*q + (1-p)*(1-q)))
}

bayes(0.01, 0.8)
```

```
## [1] 0.03883495
```

Reliable Witnesses with fixed probability

In this part, we will assume that we are using testimony of reliable witnesses with fixed reliability p.

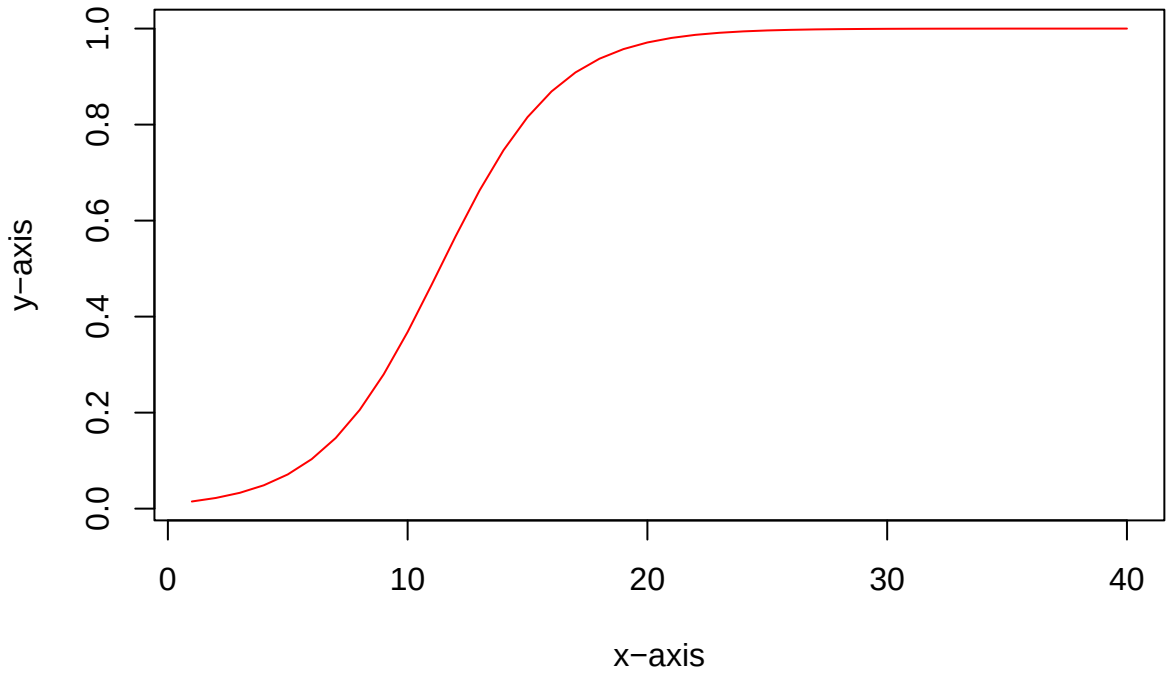
Initial Plots

Initialize $p = 0.6$, $q = 0.01$, and q_vals = c(), the empty vector. Run a for loop (1 in 1:40) inside it, update q at every iteration to $q = \text{bayes}(p, q)$, and concatenate the vector q_vals with this new value of q.

```
p = 0.6
q = 0.01
q_vals = c()

for (x in 1:40){
  q = bayes(p,q)
  q_vals <-c(q_vals, q )
}
```

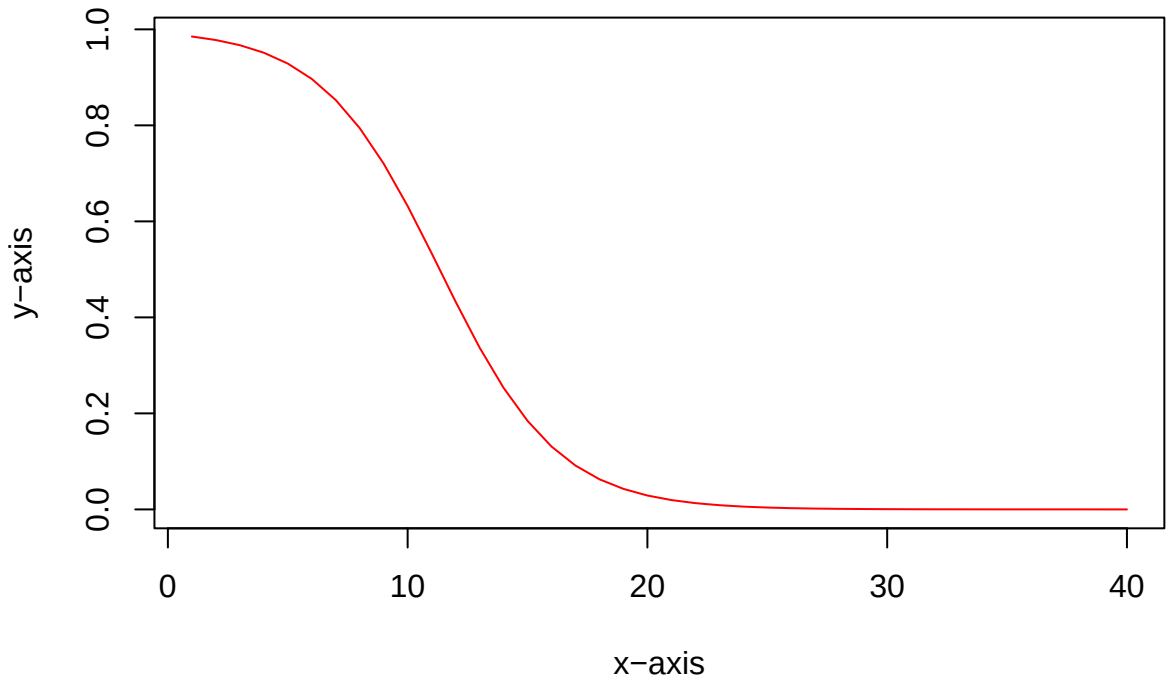
Construct a plot with x-axis 1:40 and y-axis q_vals.

```
plot(1:40,q_vals,type="l",col="red",xlab="x-axis", ylab="y-axis")
```



Repeat the above experiment with $p = 0.4$, $q = 0.99$.

```
p = 0.4
q = 0.99
q_vals = c()

for (x in 1:40){
  q = bayes(p,q)
  q_vals <-c(q_vals, q )
}

plot(1:40,q_vals,type="l",col="red",xlab="x-axis", ylab="y-axis")
```



How many reliable Witnesses needed?

Suppose the police decide to arrest the Blue cab driver if $P(B|W) \geq 0.9$.

Suppose we intialize $p = 0.6$ (witness reliability), $q = 0.01$ (proportion of Blue cabs in the city), how many witnesses with reliability level p would we need for the police to consider arresting the Blue cab?

You will need a counter variable initialized to zero and a while loop inside which you update q using the bayes function, and update the counter by 1 at every iteration of the while loop.

```
count = 0
p = 0.6

q = bayes(0.6,0.01)
```

```
    while(q < 0.9 ){
      q = bayes(p,q)
      count = count+1
    }
print("You need this amount of witnesses")
```

```
## [1] "You need this amount of witnesses"
```

```
print(count)
```

```
## [1] 16
```

Working with two types of witnesses

In this problem we will work with two types of witnesses (1) reliable witnesses with fixed reliability level $p_1 > \frac{1}{2}$ and (2) unreliable witness with fixed reliability level $p_2 < \frac{1}{2}$.

Initial Plots

Initialize $p_1 = 0.6$, $p_2 = 0.45$, $q = 0.01$, and q_vals = c(), the empty vector. Run a for loop (1 in 1:90) inside it, update q at every even iteration to $q = \text{bayes}(p_2, q)$, and at every odd iteration to $q = \text{bayes}(p_1, q)$ (You will need to use a "if" command), and finally concatanating the vector q_vals with the new value of q at every iteration.

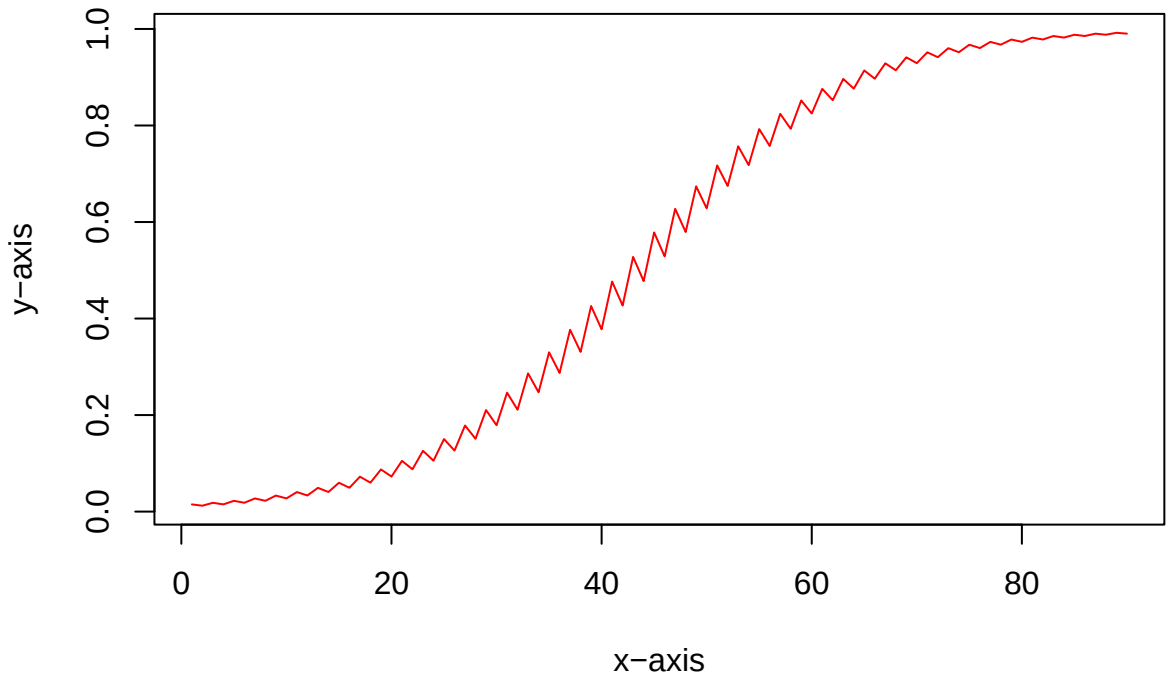
```
p_1 = 0.6
p_2 = 0.45
q = 0.01
q_vals = c()

for(x in 1:90){

  if(x %% 2 == 0){
    q = bayes(p_2,q)
  } else {
    q = bayes(p_1,q)
  }
  q_vals = c(q_vals, q)


}

print(q_vals)
```

```
##  [1] 0.01492537 0.01224490 0.01825558 0.01498612 0.02231200 0.01832963
##  [7] 0.02724475 0.02240213 0.03323098 0.02735426 0.04047776 0.03336371
## [13] 0.04922442 0.04063823 0.05974341 0.04941777 0.07233924 0.05997548
## [19] 0.08734397 0.07261645 0.10510838 0.08767325 0.12598704 0.10549689
## [25] 0.15031638 0.12644182 0.17838506 0.15084383 0.21039718 0.17899025
## [31] 0.24643100 0.21108307 0.28639775 0.24719759 0.33000781 0.28724128
## [37] 0.37675248 0.33092021 0.42590932 0.37772126 0.47657554 0.42691792
```

```
## [43] 0.52772850 0.47760430 0.57830579 0.52875614 0.62729197 0.57931110
## [49] 0.67379748 0.62825558 0.71711699 0.67470320 0.75676045 0.71795278
## [55] 0.79245613 0.75751871 0.82413081 0.79313354 0.85187499 0.82472770
## [61] 0.87590146 0.85239458 0.89650421 0.87634900 0.91402226 0.89688620
## [67] 0.92881060 0.91434577 0.94121890 0.92908278 0.95157718 0.94144663
## [73] 0.96018737 0.95176684 0.96731913 0.96034471 0.97320900 0.96744924
## [79] 0.97806145 0.97331631 0.98205123 0.97814976 0.98532630 0.98212377
## [85] 0.98801107 0.98538580 0.99020951 0.98805982 0.99200807 0.99024941
```

Construct a plot with x-axis 1:90 and y-axis q_vals.

```
plot(1:90,q_vals,type="l",col="red",xlab="x-axis", ylab="y-axis")
```



What do you notice? (Compared to case with single witness)

It takes more witnesses to reach a higher level of confidence

How many witnesses needed?

Suppose the police decide to arrest the Blue cab driver if $P(B|W) \geq 0.9$.

Initialize $p_1 = 0.6$, $p_2 = 0.45$, $q = 0.01$. How many witnesses will we have to have if we are going to sequentially work with one reliable witness followed by an unreliable witness?

```
p_1 = 0.6
p_2 = 0.45
```

```
q = 0.01
count = 1

while(q<0.9){

  if(count %% 2 == 0){
    q = bayes(p_2,q)
  } else {
    q = bayes(p_1,q)
  }
  count = count + 1

}

print("We need this many witnesses")
```

```
## [1] "We need this many witnesses"
```

```
print(count-1)
```

```
## [1] 65
```

This problem can be generalized to choosing p from any given probability distribution on $(0, 1)$, we will be working with the Beta distribution later in the semester, which can be an interesting candidate to sample our values for p (that is witness reliability).

Working with four types of witnesses

In this problem we will work with two types of witnesses (1) two reliable witnesses with fixed reliability level $p_1, p_2 > \frac{1}{2}$ and (2) two unreliable witnesses with fixed reliability level $p_3, p_4 < \frac{1}{2}$.

Initial Plots

Initialize $q = 0.01$ and we will sample p from the vector c(0.6, 0.8, 0.45, 0.3) and suppose choosing a witness with either of these witness-reliabilities is equally likely.

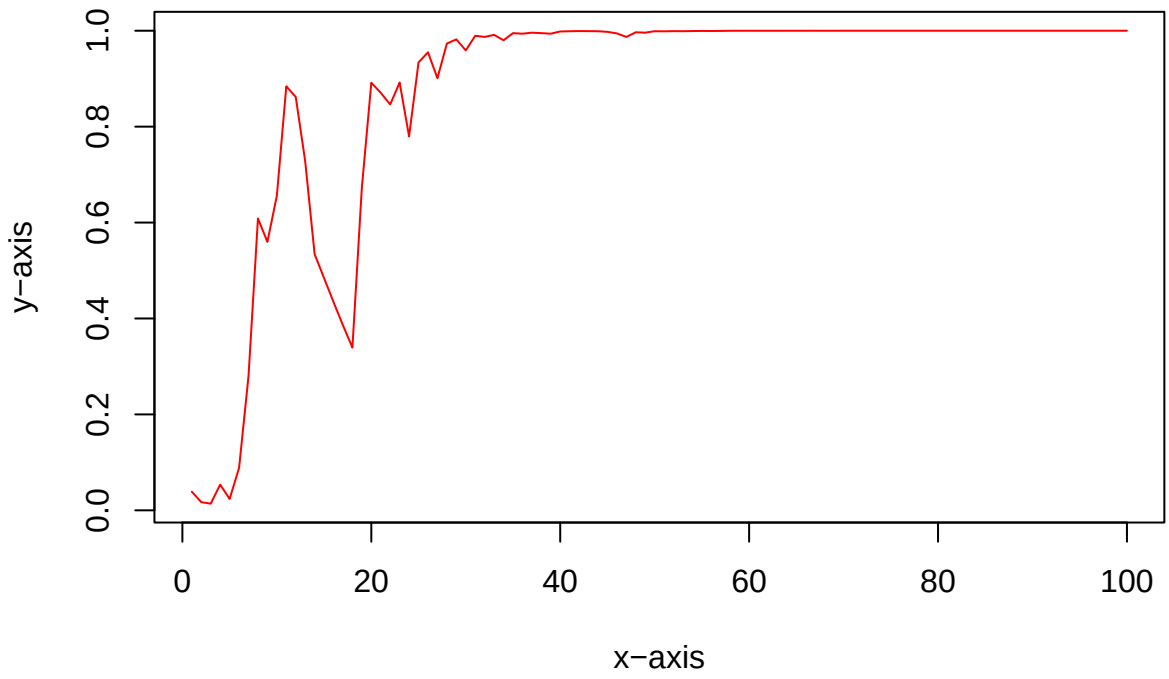
Now let q_vals = c(), the empty vector and run a for loop (1 in 1:100) inside it, update q by sampling p from the vector c(0.6, 0.8, 0.45, 0.3), and concatanating the vector q_vals with the new value of q at every iteration.

```
    q = 0.01
    q_vals = c()

    for(x in 1:100){
      q = bayes(sample(c(0.6, 0.8, 0.45, 0.3), 1), q)
      q_vals = c(q_vals, q)
    }

    print(q_vals)
```

```
##   [1] 0.03883495 0.01702128 0.01396973 0.05363128 0.02371151 0.08854727
##   [7] 0.27984939 0.60851766 0.55981588 0.65608142 0.88413400 0.86194075
##  [13] 0.72794193 0.53417361 0.48406470 0.43427417 0.38577567 0.33944356
##  [19] 0.67272140 0.89156369 0.87058523 0.84624812 0.89196171 0.77965202
##  [25] 0.93400693 0.95501505 0.90097466 0.97325753 0.98201133 0.95900959
##  [31] 0.98942736 0.98710818 0.99136836 0.98008866 0.99494670 0.99383068
##  [37] 0.99587864 0.99496739 0.99385591 0.99845687 0.99897071 0.99931357
##  [43] 0.99916116 0.99897495 0.99761147 0.99444446 0.98713239 0.99675175
##  [49] 0.99603278 0.99900524 0.99878444 0.99918930 0.99900932 0.99933933
##  [55] 0.99955946 0.99946161 0.99964101 0.99991023 0.99997756 0.99997257
##  [61] 0.99993599 0.99992177 0.99998044 0.99998696 0.99996958 0.99997972
##  [67] 0.99999493 0.99998817 0.99997239 0.99999310 0.99999540 0.99999693
##  [73] 0.99999923 0.99999981 0.99999977 0.99999994 0.99999986 0.99999983
##  [79] 0.99999980 0.99999952 0.99999988 0.99999992 0.99999995 0.99999996
##  [85] 0.99999999 0.99999999 0.99999997 0.99999994 0.99999996 0.99999999
##  [91] 0.99999999 0.99999997 0.99999998 1.00000000 1.00000000 1.00000000
##  [97] 1.00000000 1.00000000 1.00000000 1.00000000
```

Construct a plot with x-axis 1:100 and y-axis q_vals.

```
plot(1:100,q_vals,type="l",col="red",xlab="x-axis", ylab="y-axis")
```



What do you notice? How does this compare with the earlier cases?

The amount of witnesses increases faster than the earlier cases around the point 22.

How many witnesses needed?

Suppose the police decide to arrest the Blue cab driver if $P(B|W) \geq 0.9$.

Initialize $q = 0.01$ and we will sample p from the vector c(0.6, 0.8, 0.45, 0.3) and suppose choosing a witness with either of these witness-reliabilities is equally likely. How many witnesses will we need to have if we are going to uniformly sample from each of these four types of of witnesses at every step?

```
count = 0
p = 0.6

q = bayes(sample(c(0.6, 0.8, 0.45, 0.3), 1),0.01)

while(q < 0.9 ){
  q = bayes(sample(c(0.6, 0.8, 0.45, 0.3), 1),q)
  count = count+1
}

print("You need this many witnesses")
```

```
## [1] "You need this many witnesses"
```

```
print(count)
```

```
## [1] 15
```

BONUS: Working with witnesses coming from a Beta distribution

In this problem we will assume that the witness reliability of the people in the city follows the Beta distribution with parameters α, β. Choose α, β in such a way that the expected value of this Beta distribution is greater than .65.

Initial Plots

Initialize $q = 0.01$ and we will choose p by sampling the Beta distribution with parameters α, β chosen above.

Now let q_vals = c(), the empty vector and run a for loop (1 in 1:100) inside it, update q by sampling p Beta(α, β), and concatanating the vector q_vals with the new value of q at every iteration.

Construct a plot with x-axis 1:100 and y-axis q_vals.

What do you notice? How does this compare with the earlier cases?

How many witnesses needed?

Suppose the police decide to arrest the Blue cab driver if $P(B|W) \geq 0.9$.

Initialize $q = 0.01$ and we will choose p by sampling the Beta distribution with parameters α, β chosen above. How many witnesses will we need to have in this setting?