Chapter 3

Classic Problems

In this Chapter we will compile some famous problems. We will discuss the problem as originally stated and give the solution. We will consider different applications of the problems in Computer Science and Engineering.

3.1 The Birthday Paradox

What is the probability that in a group of k people at least two people share the same birthday?
Assumptions:
1. There are 356 days, i.e., we are not considering leap year.
2. The birthday of a person is equally likely to be any of the 365 days. Data shows that this not really true.
3. We assume that the births are independent. The birth of one individual does not influence the birth of another person. Again this is not really true if you consider twins.
For $k > 365$, P(atleast 2 people will have the same birthday) $=$ 1. This is based on the Pigeon-hole principle.

$$P(\text{no match}) = \frac{365 \times 364 \times \ldots \times (365 - k + 1)}{365^k}$$

$$\begin{aligned} P(\text{atleast 2 people with same birthday}) &= 1 - P(\text{no match}) \\ &= 1 - \frac{365 \times 364 \times \ldots \times (365 - k + 1)}{365^k} \\ &= 50.7\% \quad \text{for k = 23} \\ &= 97.1\% \quad \text{for k = 50} \\ &= 99.99\% \quad \text{for k = 100} \end{aligned}$$

Notes:

1. Most people are surprised that with only 23 people there is a 50% chance that two people will have the same birthday. The surprise is because 23 seems like a small number compared to 365.

2. One way to reconcile is that with 23 people there are $\binom{23}{2} = 254$ different associations being tested to see if there is match. This is a large number.

3. As we will see below the probability of a match as a function of k is not a linear function.

4. It is important to note that we have found the probability that there is at least 1 match. It includes 1 match, 2 matches, 3 matches, and so on. It is not so easy to find probability of exact number of matches, say, exactly 3 matches.

5. It is important to clarify what we mean when we say 2 matches. Lets take an example:

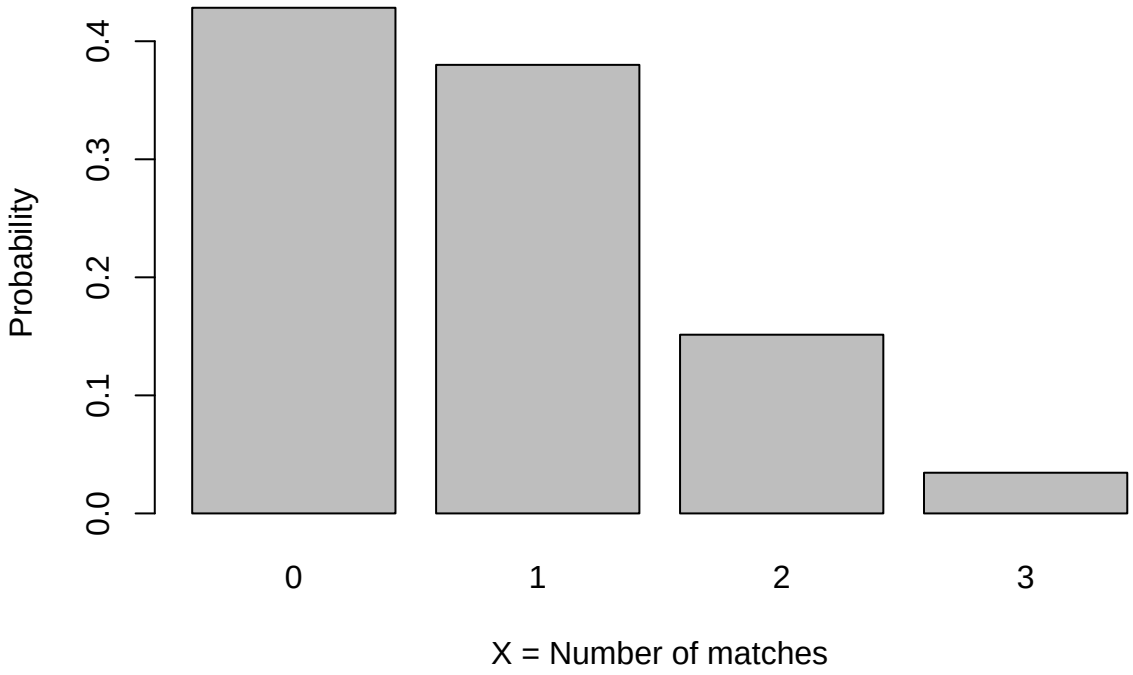
    1. If A and B have the same birthday and C and D have a different same birthday, then we have two matches. No confusion here.

    2. If A, B, and C have the same birthday, is it 2 matches or 3 matches (A and B match, A and C match, B and C match)? This is considered to be 2 matches. We are looking for unique coincidences. If A and B match and B and C match then the match A and C is not longer a coincidence.

3.1.1 R Code

We want to write R code to compute the probability of at least one match as well as exact numbwer of matches.

```
set.seed(1237)
m = 100000;                  # number of samples
n = 25                       # number of people in room change this to 50
x = numeric(m)               # vector for numbers of matches
for (i in 1:m)
{
b = sample(1:365, n, repl=T) # n random birthdays in the ith sample
x[i] = n - length(unique(b)) # no. of matches in the ith sample
}
p=numeric(4)                 # We will store the probabilities
p[1] <- mean(x == 0);                # approximates P{X=0}
p[2] <- mean(x == 1);                # approximates P(X=1)
p[3] <- mean(x == 2);                # approximates P(X=2)
p[4] <- mean(x == 3);                # approximates P(X=3)
barplot(p, names.arg=c(0:3), ylab="Probability", xlab="X = Number of matches")         # This gives the
```




```
# You can draw he entire histogram using  the following commands

#cutp = (0:(max(x)+1)) - .5   # break points for histogram
#hist(x, breaks=cutp, prob=T) # relative freq. histogram
```

3.1.2 Application 1: Saving Power in a Wireless Sensor Network

This problem is related to the application of the birthday paradox problem in sensor networks. We discussed this problem in the lecture and in one of the assignment problems. The problem relates to synchronizing 2 sensor nodes while minimizing sensor battery power usage. We consider time is divided into fixed length frames with each frame containing n fixed length slots. In each frame, each sensor node independently and randomly selects m slots to be awake (and sleeps the remaining $n - m$ slots). Let $P(\text{match})$ denote the probability that in a frame there is at least one common slot in which both the sensors are awake. We want to find $P(\text{match})$

1. If $m > n/2$ then the $P(\text{match}) = 1$. This makes sense, since if each node selects greater than half of of the number of slots then, there will be atleast one match.

2. If $m \leq n/2$ then two sensor nodes will not have a single slot in common is given by

$$P(\text{no match}) = \frac{\binom{n}{m}\binom{n-m}{m}}{\binom{n}{m}^2}$$

Thus,

$$P(\text{match}) = 1 - P(\text{no match})$$

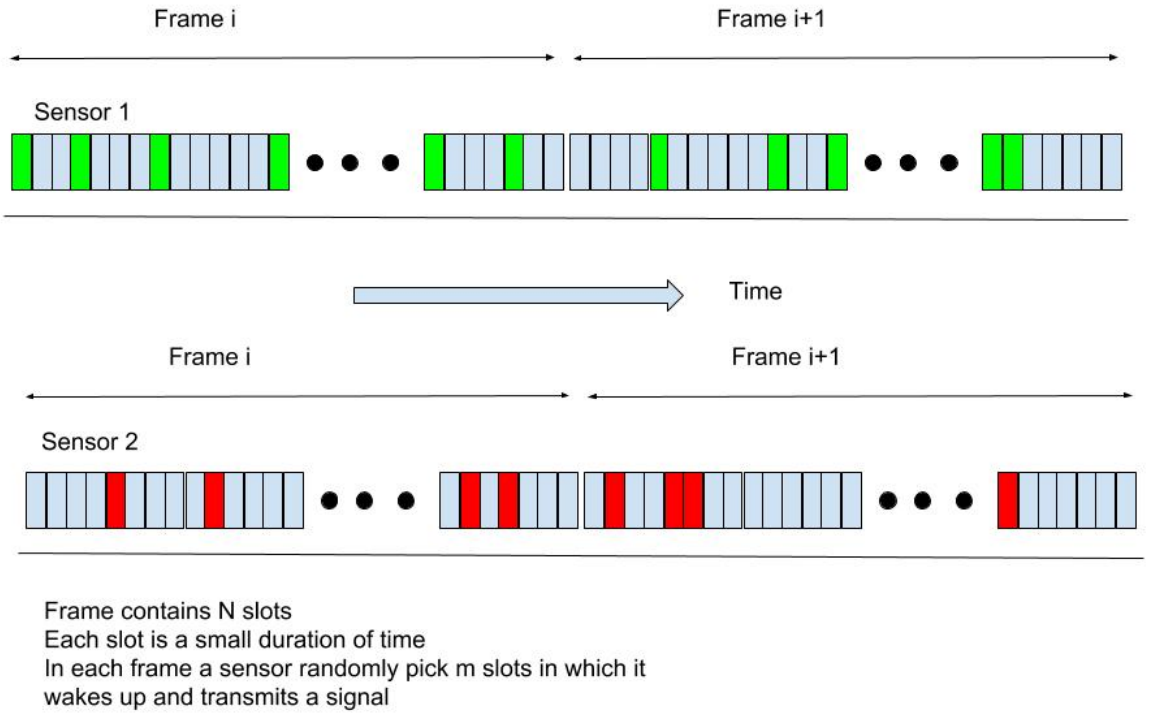


Figure 3.1: This figure illustrate the frame structure and the slots.

3.2 Matching Problem

There is a deck of n cards labelled $1, \ldots, n$. The cards are shuffled. Call out numbers 1 through n in sequence and for each call flip a card from the deck. What is the probability that for at least one called number, the corresponding card number will match? This problem was first considered by Pierre-Remond Montemort's problem around 1713.

Another version of the problem is called the Hat matching problem. Suppose n people throw their hats in a room. Then they randomly pick a hat. What is the probability that at least one person picks his own hat.

Let A_j be the event that the jth card matches.

$$\begin{aligned} P(\text{at least 1 match}) &= P(A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_n) && (3.1)\\ &= \sum_{i=1}^{i=n} P(A_i) - \sum_{i_1<i_2} P(A_{i_1} \cap A_{i_2}) && (3.2)\\ &+ (-1)^{r+1} \sum_{i_1<i_2<\ldots<i_r} P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_r}) + \ldots && (3.3)\\ &+ (-1)^{n+1} P(A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_n) && (3.4) \end{aligned}$$

Now

$$P(A_j) = \frac{1}{n} \qquad \text{for all } j = 1, \ldots, n \tag{3.5}$$

This comes form the observation that all positions are equally likely for card labelled j. Furthermore,

$$P(A_i \cap A_j) = \frac{(n-2)!}{n!} \tag{3.6}$$

This is based on the observation that 2 cards A_i and A_j are in the right positions so the remaining $n-2$ can be permuted in $(n-2)!$ ways which gives the total number of favorable outcomes out of $n!$ which is the total number of outcomes. This can be generalized to

$$P(A_i \cap A_j \ldots \cap A_k) = \frac{(n-k)!}{n!} \tag{3.7}$$

$$\begin{aligned} P(\text{at least 1 match}) &= n \times \frac{1}{n} + \binom{n}{2} \times \frac{(n-2)!}{n!} + \ldots + \binom{n}{k} \times \frac{(n-k)!}{n!}\\ &+ (-1)^{n+1} \binom{n}{n} \times \frac{(n-n)!}{n!}\\ &= \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} + \ldots + (-1)^{n+1} \frac{1}{n!}\\ &\approx 1 - \frac{1}{e} \end{aligned}$$

Couple of notes:

1. Note that the number of terms taking k events at a time is $\binom{n}{k}$.
2. To get the final result we use the asymptotic value of the Taylor series as $n \to \infty$.

3. As $n \to \infty$, the chance of getting a match becomes smaller and smaller however, the number of attempts for a match becomes larger and larger. These two opposite forces converges to $1 - \frac{1}{e}$.

3.3 Coupon Collector Problem

The following problem is called the coupon collector problem and has many applications in computer science. There are N different types of coupons. Each time one obtains a coupon it is independent of the previous selection and equally likely to be any of the N types. Let T denote the random variable that denotes the coupons that needs to be collected until one obtains a complete set of atleast one of each type.

Derivation of the expected number of coupons required to be drawn to collect the complete set. Let T denote the number of coupon required to be drawn to collect the complete set. T is a random variable and we need to find $E(T)$. If we define T_j to be the number of additional coupons that needs to be drawn to collect the jth disticnt coupon, then $T_j \sim \text{Geom}(p_j)$ where $p_j = \frac{n-j+1}{n}$. And $E(T_j) = \frac{1}{p_j}$.

$$\begin{aligned} E(T) &= E(T_1 + T_2 + \ldots + T_n) \\ &= E(T_1) + E(T_2) + \ldots + E(T_n) \\ &= 1 + \frac{n}{n-1} + \frac{n}{n-2} + \ldots + \frac{n}{1} \\ &= n \underbrace{(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n})}_{\text{Harmonic Series}} \\ &\approx n\ log(n) + 0.5772\ n + 0.5 \end{aligned}$$

3.4 Gambler's Ruin Problem

Two players A and B play a sequence of rounds; in each round they bet \$1. If A wins the round, he gets \$1 from B, if he loses, he gives \$1 to B. The probability that A wins a round is p and hence the probability that he loses a round is $q = 1 - p$. Suppose A and B combined have $\$N$ of which A has $\$i$ and B $\$N - i$ and they continue to play until A wins the game (and hence B becomes bankrupt) or A loses the game (hence A becomes bankrupt). In class we derived the probability P_i that A wins the game starting with \$i.

The basic strategy is to condition on the first step and using the Law of Total Probability

$$P_i = pP_{i+1} + qP_{i-1} \qquad 1 \leq i \leq N-1$$

with $P_0 = 0$ and $P_N = 1$.

Since $p + q = 1$, we can rewrite the above equation as

$$\begin{aligned} (p+q)P_i &= pP_{i+1} + qP_{i-1} \qquad 1 \leq i \leq N-1 \\ pP_{i+1} - pP_i &= qP_i - qP_{i-1} \\ P_{i+1} - P_i &= \frac{q}{p}(P_i - P_{i-1}) \end{aligned}$$

For $i = 1$, the equation can be written as:

$$\begin{aligned} P_2 - P_1 &= \frac{q}{p}(P_1 - P_0) \\ P_2 - P_1 &= \frac{q}{p}P_1 \end{aligned}$$

Similarly for $i = 2$ we can easily show that:

$$\begin{aligned} P_3 - P_2 &= \frac{q}{p}(P_2 - P_1) \\ P_3 - P_2 &= (\frac{q}{p})^2 P_1 \end{aligned}$$

and for $i = N - 1$

$$\begin{aligned} P_N - P_{N-1} &= \frac{q}{p}(P_{N-1} - P_{N-2}) \\ P_N - P_{N-1} &= (\frac{q}{p})^{N-1} P_1 \end{aligned}$$

Now if we take the first $i - 1$ terms and sum them we will get:

$$P_i - P_1 = P_1[\frac{q}{p} + (\frac{q}{p})^2 + \ldots + (\frac{q}{p})^{i-1}]$$

We can show that:

$$P_i = \begin{cases} \frac{1-(\frac{q}{p})^i}{1-(\frac{q}{p})} P_1 & \text{if } \frac{q}{p} \neq 1 \\ iP_1 & \text{if } \frac{q}{p} = 1 \end{cases}$$

Considering $\frac{q}{p} \neq 1$ and using the fact $P_N = 1$, we have

$$\begin{aligned} P_N &= 1 \\ &= \frac{1 - (\frac{q}{p})^N}{1 - (\frac{q}{p})} P_1 \end{aligned}$$

From which we obtain:

$$P_1 = \frac{1 - (\frac{q}{p})}{1 - (\frac{q}{p})^N}$$

Similarly, if $\frac{q}{p} = 1$, then $P_N = 1 = NP_1$ which implies that $P_1 = \frac{1}{N}$ and thus $P_i = \frac{i}{N}$, now we have:

$$P_i = \begin{cases} \frac{1-(\frac{q}{p})^i}{1-(\frac{q}{p})^N} & \text{if } \frac{q}{p} \neq 1 \\ \frac{i}{N} & \text{if } \frac{q}{p} = 1 \end{cases}$$

3.4.1 Notes and Remarks

1. We first show that $p = q$ is a limiting case of $p \neq q$. Let $x = \frac{q}{p}$. We want to find the limit of P_i as $x \to 1$.

$$\begin{aligned} \lim_{x \to 1} P_i &= \lim_{x \to 1} \frac{1 - x^i}{1 - x^N} \\ &= \frac{i}{N} \end{aligned}$$

2. Does the game go forever? Consider the case with $\frac{q}{p} = 1$. $P(\text{A wins starting with } i)) = P_i = \frac{i}{N}$. Similarly, $P(\text{A loses starting with } i) = P(\text{B wins starting with } N - i) = \frac{N-i}{N}$. Since the probability of A winning or B winning is equal to 1, then the probability of neither winning (game going forever) is 0.

3. Let $i = N - i$, i.e., $i = \frac{N}{2}$. Also let $p = 0.49$

   3.1. $N = 20 \Rightarrow P_i = 0.4$

   3.2. $N = 100 \Rightarrow P_i = 0.12$

   3.3. $N = 200 \Rightarrow P_i = 0.02$

4. If $p < q$ which implies $x = \frac{q}{p} > 1$, then

$$\begin{aligned} \lim_{N \to \infty} P_i &= \lim_{N \to \infty} \frac{1 - x^i}{1 - x^N} \\ &= 0 \end{aligned}$$

This implies that with probability 1, the Gambler will get "ruined".

3.5 Hashing

A hash table is a very important data structure in computer science. It is used for fast information retrieval. It stores data as a $< key,\ value >$ pair, where the value is indexed by the key. Note that keys must ne unique. Consider the example of storing persons name using the social security number (ssn) as the key. For each ssn x, a hash function h is used, where $h(x)$ is the location to store the name of x. Once we have created a table, to look up the name for ssn x, we can recomputes $h(x)$ and then look up what is stored in that location. In Python, dictionaries are based on hash tables. Typically, the hash function h is deterministic; we do not want to get different results every time we compute $h(x)$. But h is often chosen to be pseudo-random. For this problem, we will assume that h is truly random. Suppose there are k people, with each person' s name stored in a random location (independently), represented by an integer between 1 and n, $k < n$. It may happen that one location has more than one name stored there, if two different people ssns x and y end up with the same random location for their name to be stored.

1. What is the expected number of locations with no name stored?

2. What is the expected number with exactly one name?

3. What is the expected number with more than one name?

Answer

Let I_j be an indicator random variable equal to 1 if the jth location is empty, and 0 otherwise, for $1 \leq j \leq n$. Then $P(I_j = 1) = (1 - ĹŠ1/n)^k$, since the names are stored in independent random locations. Then $I_1 + \ldots + I_n$ is the number of empty locations. By linearity of expectation, we have

$$\begin{aligned} E(\sum_{j=1}^{n} I_j) &= \sum_{j=1}^{n} E(I_j) \\ &= n(1 - \frac{1}{n})^k \end{aligned}$$

Similarly, the probability of a specific location having exactly 1 name stored is $\frac{k}{n}(1 - \frac{1}{n})^{k-1}$. How do we get this? Consider a particular location. Any name is mapped to that location with probability $\frac{1}{n}$ and not mapped to that location with probability $1-\frac{1}{n}$. Let X be a random variable that denotes the number of names that are mapped to that location. Clearly, X can takes values $\{0, 1, 2, 3, \ldots, k$. Clearly, $X \sim \text{Binom}(k, \frac{1}{n})$. Hence,

$$\begin{aligned} P(X = 1) &= \binom{k}{1}(\frac{1}{n})(1 - \frac{1}{n})^{k-1} \\ &= \frac{k}{n}(1 - \frac{1}{n})^{k-1} \end{aligned}$$

Hence, the expected number of such locations is $k(1 - 1/n)^{k-1}$. This is again obtained by using indicator random variables. Consider, and I_i be an indicator random variable which is defined as follows

$$I_i = \begin{cases} 1 & \text{If location i has exactly one name mapped} \\ 0 & \text{otherwise} \end{cases}$$

Let Y be the random variable that denotes the number of locations with exactly one name. Then

$$\begin{aligned} E(Y) &= E(I_1 + I_2 + \ldots + I_k) \\ &= E(I_1) + E(I_2) + \ldots + E(I_k) \quad \text{by Linearity of expectation} \\ &= nP(I_1 = 1) \quad \text{by symmetry} \\ &= n\frac{k}{n}(1 - \frac{1}{n})^{k-1} \\ &= k(1 - \frac{1}{n})^{k-1} \end{aligned}$$

Finally, the number of locations with more than 1 name is $n - n(1 - \frac{1}{n})^k - k(1 - \frac{1}{n})^{k-1}$.