**Question:** What is the probability that we toss N balls into M bins, and each bin ends up with exactly K balls?

**Answer:** Let’s start with a simple example – N=4, M=2, K=2. One way to think about this is to use a tree diagram, shown to the right. Each arc labeled “0” represents a ball landing in bin #0; each arc labeled “1” represents a ball landing in bin #1. We can then label each possible event with a sequence of ones and zeros as shown. For example, “0001” means three balls land in bin #0 and one ball lands in bin “1”. This is beginning to look like a Digital Communications problem :)



As you can see from the diagram, there are 24, or MN in the general case, possible outcomes for this event. The table below summarizes all possible events and how many balls end up in each bin:

|  |  |  |
| --- | --- | --- |
| **No.** | **Balls** | **Bins** |
| **Ball #1** | **Ball #2** | **Ball #3** | **Ball #4** | **Bin #0** | **Bin #1** |
| 1 | 0 | 0 | 0 | 0 | 4 | 0 |
| 2 | 0 | 0 | 0 | 1 | 3 | 1 |
| 3 | 0 | 0 | 1 | 0 | 3 | 1 |
| 4 | 0 | 0 | 1 | 1 | 2 | 2 |
| 5 | 0 | 1 | 0 | 0 | 3 | 1 |
| 6 | 0 | 1 | 0 | 1 | 2 | 2 |
| 7 | 0 | 1 | 1 | 0 | 2 | 2 |
| 8 | 0 | 1 | 1 | 1 | 1 | 3 |
| 9 | 1 | 0 | 0 | 0 | 3 | 1 |
| 10 | 1 | 0 | 0 | 1 | 2 | 2 |
| 11 | 1 | 0 | 1 | 0 | 2 | 2 |
| 12 | 1 | 0 | 1 | 1 | 1 | 3 |
| 13 | 1 | 1 | 0 | 0 | 2 | 2 |
| 14 | 1 | 1 | 0 | 1 | 1 | 3 |
| 15 | 1 | 1 | 1 | 0 | 1 | 3 |
| 16 | 1 | 1 | 1 | 1 | 0 | 4 |

The rows shaded green are the ones we are interested in for the case K=2. I can redraw this table by combining rows so that I can compute the probability of each outcome:

|  |  |  |
| --- | --- | --- |
| **No.** | **Bins** | **Prob.** |
| **Bin #0** | **Bin #1** |
| 1 | 4 | 0 | 1/16 = 0.0625 |
| 2 | 3 | 1 | 4/16 = 0.2500 |
| 3 | 2 | 2 | 6/16 = 0.375 |
| 4 | 1 | 3 | 4/16 = 0.2500 |
| 5 | 0 | 4 | 1/16 = 0.0625 |

How can we generalize this result?

Dr. Obeid has offered this nice explanation (for the case N=10, M=5, K=2):

“I think the easiest way to think about it is that the first box has a possibility of (10 choose 2) \* (1/5)^2 of getting two balls. The second box has a probability of (8 choose 2) \* (1/5)^2 and so on down the line. If you multiply those together, you get the right answer.”

We can write this mathematically as:



For N=4, M=2, K=2, we get:



The product has a closed-form expression (the derivation of this is beyond the scope of this course):



 Again, for N=4, M=2, K=2, we have:



There are many variants of these problems (e.g., each bin has K or more balls, what is the average number of balls in a bin). The equations are similar but of course are different depending on the specifics of the problem.

Dr. Obeid, who is our resident MATLAB guru, passes along two very nice pieces of MATLAB code:

(1) A simulation of the experiment:

This code proves experimentally that the above equations are correct, as is the textbook.

clear;

nTrials = 1e5;

outcome = zeros(1,nTrials);

for i = 1:nTrials

    h = rand(1,10);

    nb1 = sum(h<=.2);

    nb2 = sum(h<=.4) - nb1;

    nb3 = sum(h<=.6) - nb1 - nb2;

    nb4 = sum(h<=.8) - nb1 - nb2 - nb3;

    nb5 = sum(h<= 1) - nb1 - nb2 - nb3 - nb4;

    if (nb1==2) && (nb2==2) && (nb3==2) && (nb4==2) && (nb5==2)

        outcome(i) = 1;

    end

end

disp(sum(outcome)/nTrials)

(2) Calculation of the probability (for the case N=10, M=2, K=2):

clear; clc;

ncr = @(n,k) nchoosek(n,k);

p = prod([ncr(10,2) \* 0.2^2 ,...

    ncr(8,2) \* 0.2^2 ,...

    ncr(6,2) \* 0.2^2 ,...

    ncr(4,2) \* 0.2^2 ,...

    ncr(2,2) \* 0.2^2 ]);

disp(p);

This type of problem has been extensively studied in the literature over the years. It goes by various names. Google search “balls and bins probability” to find some nice documents explaining the general theory.

We use this type of analysis for engineering problems involving applications like quality control (“how many components are defective?”), reliability (“what is the probability that two or more subsystems will fail?”, load balancing (“how many web servers do I need to service N users?”) and general occupancy calculations (“how many hotel rooms do I need to achieve an 80% occupancy rate?”). We won’t dwell on these problems in this course, but it is good to have some familiarity with them.

Finally, I cannot stress the importance of working small examples manually before you jump into these generalized solutions. This is a recurring theme in my classes – you must debug your code and your problem solutions with several test cases that fully exercise the code or equations.

Also, why we are at it, you can see that almost any homework solution can require a significant amount of explanation. This is why I want to see more detailed homework solutions from you.

Enjoy!