GRAD C23: Mathematics for Data Science
Hertie School
Fall 2023
Prof. Magazinnik

Problem Set 1

Submitted by: *Giorgio Coppola*
Group: Group 3: Giorgio Coppola, Isabella Urbano Trujillo, Lino Hans Julian Zurmuehl

1. **Conducting a Survey**.
   Summary: 600 total members. 400 students, 50 faculty, 150 staff.

   a. Random sample of 60 individuals without replacement: choosing an individual does not preclude being chosen again.
   We have a sample space of 600^{60} elements, representing the total combinations (outcomes) in general, given 600 total members (n) and 60 choices (k). This represents the denominator of our probability function, since we are sampling with replacement. Indeed, the probability function will look like this:

$$P(\text{All 60 are unique}) = 1 \times \frac{599}{600} \times \frac{598}{600} \times \ldots \times \frac{541}{600} \approx 0.0472$$

   The solution to this probability function is 0.0472, namely 4.72%. Indeed, the numerator represents the number of favorable outcomes (the unique combinations), that divided by the number of total outcomes (the sample space), gives the probability value of having the favourable outcome.

   b. To answer to this second point, we need to use the binomial coefficient, which counts the number of subsets of a certain size for a certain set. It answer directly to the question "How many ways are there to construct a sample of 60 given a set of 600, if we do not care about the order?"

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\ldots(n-k+1)}{k!}$$

   namely:

$$\binom{600}{60} = \frac{600!}{60!(600-60)!} = \frac{600(600-1)\ldots(600-60+1)}{60!} \approx 2.77 \times 10^{83}$$

   The number of ways to select a sample is approximately 2.77×10^{83}.

   c. The probability to select only students is intuitively very small, even though students are the biggest part of the population. Again, the probability of only picking students is the ratio of the number of favorable outcomes to the number of possible outcomes (sample space). Therefore, deriving from the hypergeometric distribution:

$$\Pr(z = k) = \frac{\binom{K}{k} \cdot \binom{N-K}{n-k}}{\binom{N}{n}}$$

   Where:

- N is the total number of individuals (everyone) and is equal to 600.
- K is the number of students and is equal to 400.
- $N - K$ is the number of non-students and is equal to 200.
- n is 60.

Given $k = 60$:

$$\Pr(z = 60) = \frac{\binom{400}{60} \cdot \binom{200}{60-60}}{\binom{600}{60}}$$

Which simplifies to:

$$\Pr(z = 60) = \frac{\binom{400}{60} \cdot \binom{200}{0}}{\binom{600}{60}}$$

And since $\binom{200}{0} = 1$:

$$\Pr(\text{All 60 are students}) = \frac{\binom{400}{60}}{\binom{600}{60}} \approx 5.44 \times 10^{-12}$$

The value is extremely low.

d. To determine the number of ways to construct the survey sample using the blocking technique, we'll compute the number of combinations for each group and then multiply the results.

$$\binom{400}{40} \times \binom{50}{5} \times \binom{150}{15} \approx 6.78 \times 10^{81}$$

This number is significantly large. It represents the number of potential unique samples without replacement one could draw even when using a blocking technique to ensure equal representation. Remember that sampling possibilities without blocking technique were 2.77×10^{83}, while with blocking technique is 6.78×10^{81}.

e. To solve this question, we have to do a combination to select a sub-sample of 10 people from a sample of 600, namely the number of distinct possibilities to have 10 people drawn from a sample of 600. After, we need to do a permutation to know the number of possible seating outcomes.
This means:

$$\text{Total ways} = 9!\binom{600}{10} \approx 5.61 \times 10^{26}$$

Because there are $\binom{600}{10}$ ways of choosing the initial sample, and 9! ways to arrange them around the table after setting one person's position as a reference.

However, here we are assuming that the places are not numbered or that have no distinguishable features, and because of the rotatory propriety of a circle, we will have $n - 1 = 9$ arrangements, given one assigned seat. Indeed, if we number the places or we assign them some distinguishable features, even if they are arranged in a circle, we will have to arrange $n = 10$ objects, the same as people were in a straight line.
In this case we would have:

$$\text{Total ways} = 10!\binom{600}{10} \approx 5.61 \times 10^{27}$$

f. Now, we have to drawn a sub-sample of 5 from the sample of 400 students, and a sub-sample of 5 from the sample of 200 non-students, and plug these numbers as numerator of our probability function to find the (intuitively remote) probability of getting the 5 students seated all next to each other. This is a permutation, since, obviously, the position matters.
First, we have $\binom{400}{5} \times \binom{200}{5} \approx 2.11 \times 10^{20}$ number of ways to select 5 students and 5 non-students from the sample, given our assumptions.
Now we consider the students seating all next to each other. We have to notice that they can swap their seats between themselves, so the important thing is only that they seat next to each other. So, among themselves, the position does not matter. The same is for the non-student group. Therefore, there are 5! ways students can be arranged, and 6! ways non-students can be arranged. This is because when we arrange 5 students next to each other, we can treat them as a single "block" that we are placing within the 10 seats. If we consider the 5 students as one block and the 5 non-students as 5 individual entities, then we can think of the seating arrangement as placing 6 entities (1 block of students + 5 non-students) in a circle.
The probability function will be:

$$P = \frac{\text{Permutations where 5 students sit together}}{\text{Total permutations}}$$

namely:

$$P = \frac{5!\binom{400}{5} \times 6!\binom{200}{5}}{10!\binom{600}{10}} \approx 0.00325$$

if we numerate the seats (equal as if we dispose seats in a straight line).

If instead we consider to arrange people in circle on seats with no distinguishing features between seats, the number of unique arrangements is $10 - 1 = 9!$. Therefore, we will have the following function:

$$P = \frac{5!\binom{400}{5} \times 5!\binom{200}{5}}{9!\binom{600}{10}} \approx 0.00542$$

2. **Medical testing.**
Summary: prevalence of disease is 1%. Accuracy of test by company:

	Company A	Company B
Sensitivity	0.95	Higher accuracy than A
Specificity	0.95	Higher accuracy than A

a. The accuracy of a test in this population is:

$$A = (P \times D) + (N \times (1 - D))$$

where
A = accuracy of a test in the population
P = true positive test (sensitivity)
D = prevalence of disease
N = true negative test (specificity)

b. Since Company B's test is always negative, then its accuracy is higher than Company A's test. Replacing in equation (1):

$$A_B = (0 \times 0.01) + (1 \times 0.99) = 0.99$$

c. Although Company B's test is more accurate, Company A's test is still more useful because of several reasons:

- Early detection: Because its higher sensitivity, Company A's test is more likely to detect the disease in individuals who actually have it. This is crucial for early detection and timely treatment.
- False negatives: Company B's test may result in a significant number of false negatives. So, individuals with the disease might go undetected, leading to delayed treatment and potentially worse outcomes. Company A's test reduces the risk of false negatives.
- Groups with higher risk: Depending on the specific characteristics of the patient population and the disease, Company A's test might be more suitable for certain groups of patients. For example, in cases where the disease is prevalent among a particular demographic, a test with higher sensitivity may be preferred.
- Monitoring and research: Company A's test might be more suitable for disease monitoring and research purposes, where detecting all potential cases (true positives) is essential for collecting accurate data and understanding disease trends.
- Patient preferences: Patients may have preferences for a test that offers better chances of early detection, even if it carries a slightly higher risk of false positives.

d. Company A wants to beat Company B in terms if the accuracy of the test. Then,

$$A_A > 0.99$$

- Scenario 1: P = N

$$(P \times 0.01) + (P \times 0.99) > 0.99$$
$$P \times (0.01 + 0.99) > 0.99$$
$$P > 0.99$$

- Scenario 2: P = 1

$$(1 \times 0.01) + (N \times 0.99) > 0.99$$
$$0.99N > 0.98$$
$$N > 0.9899$$

- Scenario 3: N = 1

$$(P \times 0.01) + (1 \times 0.99) > 0.99$$
$$0.01P > 0$$
$$P > 0$$

3. **Monty Hall revisited.**

   a. Intuitively: For Heads we get the normal Monty Hall problem with the same probability for success P = $\frac{2}{3}$. This will happen with a probability of p. But with a probability of $(1-p)$ we get the variation. In the variation if the contestant initially chooses D_1, and Monty opens M_2 we end up with a 50% chance that we initially chose the car door or not. So the second path to success happens with a probability of $(1-p)\times\frac{1}{2}$. Application of Bayes:
   $P^{\text{v}}(C_3|M_2)$ = Probability in the Monty Hall variation that the car is behind door 3 given Monty opens door 2.

$$
\begin{aligned}
P_{\text{success}} &= (1-p)\times P^{\text{v}}(C_3|M_2) + p\times P(C_3|M_2) \\
&= (1-p)\times\frac{P^{\text{v}}(M_2|C_3)\times P^{\text{v}}(C_3)}{P^{\text{v}}(M_2)} + p\times\frac{P(M_2|C_3)\times P(C_3)}{P(M_2)} \quad \text{(by Bayes' Rule)} \\
&= (1-p)\times\frac{P^{\text{v}}(M_2|C_3)\times P^{\text{v}}(C_3)}{P^{\text{v}}(M_2|C_1)\times P^{\text{v}}(C_1) + P^{\text{v}}(M_2|C_2)\times P^{\text{v}}(C_2) + P^{\text{v}}(M_2|C_3)\times P^{\text{v}}(C_3)} \\
&\quad + p\times\frac{P(M_2|C_3)\times P(C_3)}{P(M_2)} \quad \text{(by LOTP)} \\
&= (1-p)\times\frac{\frac{1}{3}\times\frac{1}{3}}{\frac{1}{3}\times\frac{1}{3}+0\times\frac{1}{3}+\frac{1}{3}\times\frac{1}{3}} + p\times\frac{2}{3} \\
&= (1-p)\times\frac{1}{2} + p\times\frac{2}{3} \\
&= \frac{1}{2} - p\times\frac{1}{2} + p\times\frac{2}{3} \\
&= \frac{1}{2} - p\times\frac{3}{6} + p\times\frac{4}{6} \\
&= \frac{1}{2} + p\times\frac{1}{6}
\end{aligned}
$$

   The probability of success when the contestant chooses door 1, then Monty opens door 2 and the contestant switches to door 3 is $\frac{1}{2} + p\times\frac{1}{6}$.

   b. The code for b is in the R appendix. For b we tried to come up with a simulation that resembles the theory, but unfortunately we were not able to get the same results as the calculations in 3a. Our function seems to simulate the variation correctly, if you look at one go (for this, we can delete the # before every cat function to check every step and then just execute the function with a given p). But if we loop it over several thousand goes we get win probabilities between 0.26 (for p = 0.1) and 0.35 (for p = 1) which contradicts the calculation (here we need to comment out the cat function). With a p = 1 we should get 2/3 and with a p = 0 we should get 0.5.

4. **PMF practice.**

   a. The Probability Mass Function of a discrete r.v. X represents the probability that X assumes certain values x (e.g., in Bernoulli, failure or success) for each k. In this case, X represents the probability of reaching a certain level k, when we have n = 7 levels. If X = k = 7, the player succeeds in winning the game, passing all the levels.

In this case, since the value of X is the highest level that the player reaches, if we take $X = 1$, it means that the player does not pass level 1. So, the probability of X = 1 is the probability of not passing the first level, which equals to $1 - p_1$. The same reasoning applies for all the other levels, considering that to pass a level, you have to pass the one before.

Therefore, if X is the highest level, the PMF $P(X = k)$ for X can be expressed as:

$$P(X = k) = \begin{cases} (1 - p_1) & \text{if } k = 1 \\ p_1 \times (1 - p_2) & \text{if } k = 2 \\ p_1 \times p_2 \times (1 - p_3) & \text{if } k = 3 \\ p_1 \times p_2 \times p_3 \times (1 - p_4) & \text{if } k = 4 \\ p_1 \times p_2 \times p_3 \times p_4 \times (1 - p_5) & \text{if } k = 5 \\ p_1 \times p_2 \times p_3 \times p_4 \times p_5 \times (1 - p_6) & \text{if } k = 6 \\ p_1 \times p_2 \times p_3 \times p_4 \times p_5 \times p_6 & \text{if } k = 7 \end{cases}$$

b. A PMF to be valid, needs to satisfy two properties: (1.) being non-negative $P(X = k) > 0$ if $x = xj$ for some j, and $px(x) = 0$ otherwise , and (2.) it must sum to 1.

For our case, we can easily check the first requirement. Since we are talking of probabilities, each pj must be between 0 and 1 (inclusive). Since each term in the PMF is a product of probabilities or (1 - probability), each term is also non-negative.

Concerning the second requirement, to check that the total probability is 1, we should sum together the partial probabilities for each k. It is quickly clear that some terms cancels out, as we can notice if we analyze a bit the results: $1-p_1+p_1(1-p_2)+p_1p_2(1-p_3)+p_1p_2p_3(1-p4)\ldots = p_1p_2p_3p_4p_5p_6 = 1$.
In the R appendix you can find the calculation on the form of a function, assigning random number to each p given k. The result is 1.

c. If the probability of getting to the next level is the same for each level, then the PMF is simpler:

$$P(X = k) = \begin{cases} 1 - p & \text{if } k = 1 \\ p(1 - p) & \text{if } k = 2 \\ p^2(1 - p) & \text{if } k = 3 \\ p^3(1 - p) & \text{if } k = 4 \\ p^4(1 - p) & \text{if } k = 5 \\ p^5(1 - p) & \text{if } k = 6 \\ p^6 & \text{if } k = 7 \end{cases}$$

We can write the same function in a more compact way:

$$P(X = k) = \begin{cases} p^{k-1}(1 - p) & \text{if } 1 \leq k \leq 6 \\ p^6 & \text{if } k = 7 \end{cases}$$

d. According to the description, we define X as a random variable representing the number of levels a player fails before successfully beating 3 levels. We are interested in finding the probability of failing k levels before succeeding 3 times in total.
To summarize, for this distribution, let:

- r representing the fixed number of successes we are aiming to achieve. In this scenario, since we are interested in the probability of failing k levels before achieving 3 successes, we have $r = 3$;
- k denoting the variable number of failures before reaching those 3 successes. Given the game's infinite levels, k can theoretically take on any non-negative integer value, ranging from 0 (indicating the player beats three levels consecutively without any failure) to infinity (the player keeps failing levels indefinitely).

Clearly, in the extreme case that $k = 0$, X is 1, as the player beats 3 levels in a row without failing any. Conversely, in theory, if $k = infinite$, X is 0, since the player will never achieve three successes.

Starting from the PMF of a binomial distribution, which is

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k},$$

we can try to derive the PMF of our X. The binomial value indicates the probability of k successes for n trials, but we are interested into the probability of k failures before r successes.

Therefore, we can transform our PMF in:

$$P(X = k) = \binom{k+r-1}{k} p^r (1-p)^k.$$

This because $r + k$ represents the number of trials (failures + successes), and k the number of failures before the success is achieved. We use $r + k - 1$ because the last trial is assumed to be a success – indeed, we are searching for the number of failures before beating the third success. If we assume that the player stops playing when she reaches r successes (which is equal to assume that she will reach r successes at a certain point), the only case in which she doesn't reach a success is if k equals infinite – that is only a theoretical case.

e. Both the binomial distribution and the negative binomial distribution model sequences of Bernoulli trials, but they are used to answer different types of questions.

The binomial distribution gives the probability of k successes in n trials, while the negative binomial gives the probability of k failures before r given successes. Indeed, we use the negative binomial when we want to find the probability of a specified number of failures/successes occurring before a predetermined number of successes/failures are observed. Moreover, the negative binomial distribution assumes that the sequence of trials stops after the rth success.

R-Appendix

R code for Question 1:

```
### Question 1
#
# a.
p_unique <- 1
for(i in 0:59) {
  p_unique <- p_unique * (600 - i) / 600
}
round(p_unique, 4) # = 0.0472

# b.
result_b <- choose(600, 60)
result_b # = 2.774267e+83, approx 2.77e+83

# c.
result_c <- choose(400, 60) / choose(600, 60)
result_c # = 5.438e-12

# d.
result_d <- choose(400, 40) * choose(50, 5) * choose(150, 15)
result_d # = 6.779343e+81

# e.
result_e_1 <- factorial(9) * choose(600, 10)
result_e_1 # = 5.607472e+26

result_e_2 <- factorial(10) * choose(600, 10)
result_e_2 # = 5.607472e+27

# f.
result_f_1 <- choose(400, 5) * choose(200, 5)
result_f_1 # = 2.110132e+20

result_f_2 <- (factorial(5) * choose(400, 5) * factorial(5) * choose(200, 5)) / (factorial(9)
round(result_f_2, 5) # = 0.00542

result_f_3 <- (factorial(5) * choose(400, 5) * factorial(6) * choose(200, 5)) / (factorial(10
round(result_f_3, 5) # = 0.00325

result_f_4 <- (factorial(5) * choose(400, 5) * factorial(5) * choose(200, 5)) / (factorial(9)
round(result_f_4, 5) # = 0.00542
```

R code for Question 3:

```
### Question 3
#
# b.
# This a function where the contestand is always switching
```

```
mont_variation <- function(p) {
  # Simulate a coin flip with probability p
  coin_flip <- ifelse(runif(1) <= p, "Heads", "Tails")
  #cat("Coin flip result:", coin_flip, "\n")

  # Determine the game mode based on the coin flip result
  if (coin_flip == "Tails") {
    game_mode <- "variation"
  } else {
    game_mode <- "standard"
  }
  #cat("Game mode:", game_mode, "\n")

  # Define the doors
  doors <- 1:3

  # Randomly select the door behind which the car is placed
  car_door <- sample(doors, 1)
  #cat("Car is behind door:", car_door, "\n")

  # Contestant's initial choice of door
  contestant_first_door <- sample(doors, 1)
  #cat("Contestant's initial choice:", contestant_first_door, "\n")

  # Logic for standard game mode
  if (game_mode == "standard") {
    monty_door <- sample(setdiff(doors, c(car_door, contestant_first_door)), 1)
    #cat("Monty opens door:", monty_door, "\n")
    contestant_door <- sample(setdiff(doors,c(monty_door, contestant_first_door)), 1)
    #cat("Contestant's final choice:", contestant_door, "\n")
  } else if (game_mode == "variation") {
    # Logic for variation game mode
    monty_door <- sample(setdiff(doors,c(contestant_first_door)), 1)
    #cat("Monty opens door:", monty_door, "\n")
    contestant_door <- sample(setdiff(doors, c(monty_door, contestant_first_door)), 1)
    #cat("Contestant's final choice:", contestant_door, "\n")
  }

  # Determine the result of the game
  if (contestant_door == car_door) {
    result <- 1 #win
    #cat("Result: Win\n")
  } else {
    result <- 0 #loss
    #cat("Result: Loss\n")
  }

  if (monty_door == car_door) {
    result <- 0 #loss
    #cat("Monty revealed the car, Result: Loss\n")
  }
```

```
  # Return a list with game information
  return(result)
}


# Create an empty list to store the results for each p value
results_list <- list()

# Set the range of p values with 0.1 steps
p_values <- seq(0.1, 1.0, by = 0.1)

# Loop through different values of p
for (i in 1:length(p_values)) {
  p <- p_values[i]  # Get the current p value

  # Repeat the mont_variation function 10,000 times for the current p
  results_for_p <- replicate(10000, mont_variation(p))

  # Store the results in the list
  results_list[[i]] <- results_for_p
}

# Calculate the proportion of wins for each p value
proportions_wins <- sapply(results_list, function(results_for_p) mean(results_for_p))

# Print the proportions of wins for each p value
for (i in 1:length(p_values)) {
  cat("p =", p_values[i], "Proportion of wins:", proportions_wins[i], "\n")
}
```

R code for Question 4:

```
### Question 4
#
# b.
check_PMF_validity <- function(p1, p2, p3, p4, p5, p6) {
  # Calculate the probabilities for X = 1 to X = 7
  p_values <- c(1 - p1,
                p1 * (1 - p2),
                p1 * p2 * (1 - p3),
                p1 * p2 * p3 * (1 - p4),
                p1 * p2 * p3 * p4 * (1 - p5),
                p1 * p2 * p3 * p4 * p5 * (1 - p6),
                p1 * p2 * p3 * p4 * p5 * p6)

  # Sum the probabilities
  total <- sum(p_values)

  # Check if they sum to 1 (accounting for potential floating point inaccuracies)
  is_valid <- abs(total - 1) < 1e-9
```

```
  # abs compute the absolute value of x, and define if it is sufficiently closed to 1 (accoun

  return(list(sum = total, is_valid = is_valid))
}

### Test the function: giving random values to probabilities
p1 <- runif(1) # generates n random numbers between 0 and 1 from a uniform distribution
p2 <- runif(1)
p3 <- runif(1)
p4 <- runif(1)
p5 <- runif(1)
p6 <- runif(1)

# Print the random probabilities
cat("p1:", p1, "\np2:", p2, "\np3:", p3, "\np4:", p4, "\np5:", p5, "\np6:", p6, "\n")
# p1: 0.8714131
# p2: 0.2548088
# p3: 0.4642005
# p4: 0.5919258
# p5: 0.9023716
# p6: 0.3368228

# Result:
check_PMF_validity(p1, p2, p3, p4, p5, p6)

# $sum
# [1] 1

# $is_valid
# [1] TRUE
```