White Swan Data - Keno Project

Hassan Ali

September 15, 2025

Question 1:

See `Q1.py` for a Keno game simulator in Python.

Question 2:

For a list of selected numbers of length n and number of matches r, we use the hypergeometric distribution as detailed by the sources provided in the question paper. The general formula is given in Equation (1).

$$\Pr[X = r] = \frac{\binom{n}{r} \times \binom{80-n}{20-r}}{\binom{80}{20}} \tag{1}$$

Fixing $n = 10$, the results of the above computation $\forall\ r \in \{5, 6, 7, 8, 9, 10\}$ are provided in the second column Table 1. To calculate fair odds, we calculate the reciprocal of the implied probabilities. These odds are provided in the third column of Table 1 for each r value.

$\boldsymbol{r}$	$\mathbf{Pr}[\boldsymbol{X = r}]$	**Fair Odds**
5	5.14×10^{-2}	19.44
6	1.15×10^{-2}	87.11
7	1.61×10^{-3}	620.68
8	1.35×10^{-4}	7,384.47
9	6.12×10^{-6}	163,381.37
10	1.12×10^{-7}	8,911,711.18

Table 1: The number of matches, their associated probability via Equation (1) and corresponding fair odds.

These odds are fair, as the amount of money that is won is proportional to the probability of winning. For example, if we bet £1 on any odds $O \in \mathbb{R}_{\geq 0}$, if we win, we are guaranteed to receive exactly $£1 \cdot O$. As match events are mutually exclusive, no conditional dependencies exist. The calculations resulting in Table 1 can be viewed in the `Q2.py` file.

Question 3:

The expected value can be calculated to see whether or not this bet is feasible from a gambler's point of view. Using the probabilities from Table 1, and the expected value equation:

$$\mathbb{E}[P] = \sum_{r=5}^{10} P(r) \cdot \Pr[X = r] \tag{2}$$

where $P(r)$ corresponds to the payoff of r, we calculate each expected payoff per unit stake in Table 2.

$\boldsymbol{r}$	$\boldsymbol{P(r)}$	$\mathbf{Pr}[\boldsymbol{X = r}]$	$\boldsymbol{P(r) \cdot \mathbf{Pr}[X = r]}$
5	3	5.14×10^{-2}	0.15
6	15	1.15×10^{-2}	0.17
7	100	1.61×10^{-3}	0.16
8	1,000	1.35×10^{-4}	0.14
9	25,000	6.12×10^{-6}	0.15
10	2,500,000	1.12×10^{-7}	0.28

Table 2: Expected values for each winning event $r \geq 5$. EV's are rounded to 2.d.p. above, with the sum being implemented on precise values.

Using Equation (2), the total expected payoff is then $\mathbb{E}[P] = 1.06$. Since the expected value exceeds the £1 stake, the bet is feasible, producing an expected profit of £0.06. Therefore, I would play the game offered. Furthermore, if this game was offered more than once, then by the Kelly criterion, the optimal fraction of capital to invest in the bet each round will always be greater than zero. The game is then worth playing repeatedly. The calculations resulting in Table 2 and the expected payoff can be viewed in the `Q3.py` file.

Question 4:

As we are asked to insure the scenario where $r = 10$, a fair premium would be the expected amount to be paid out for ten matches per unit stake (see Table 2). Therefore, for each £1 entry, a fair premium of £0.28 can be charged. However, this would only allow for the company to cover exactly eight $r = 10$ events before going bankrupt. Clearly, the premium must at least cover the expected payout per unit stake, leading to the following condition:

$$m \geq 0.28 \tag{3}$$

Let us assume that there will be $N = 10^8$ plays of the Keno game offered, with each win insured by our company recorded as a loss $L = 2.5 \times 10^6$ taken from our initial capital $C = 20 \times 10^6$. The premium amount m must act as a buffer on our capital such that the probability of going bankrupt is below a given threshold based on the risk appetite of our company. Proceeding under the assumption that N is very large, and using the fact that the probability p of winning an $r = 10$ event is extremely low (see Table 1), we can use the Poisson distribution to represent the spread of probabilities for an arbitrary number of $r = 10$ win events. Let K be the RV corresponding to the number of wins, and the mean number of events λ be equal to the expected number of wins for N plays. We then have

$$K \sim \text{Poisson}(\lambda) \quad \text{where} \quad \lambda = p \cdot N$$

Therefore, for the q^{th} quantile (confidence level) of this distribution, we derive a secondary condition for the premium, where the total capital plus the total premium for N plays is at least as large as the minimum number of winning events k associated with our chosen quantile multiplied by the loss.

$$C + m \cdot N \geq k \cdot L$$

This leads to the second condition:

$$m \geq \frac{k \cdot L - C}{N} \tag{4}$$

Calculating the expected number of wins for $N = 10^8$ total plays, we find

$$\mathbb{E}[K] = 10^8 \cdot (1.12 \times 10^{-7}) \approx 11. \tag{5}$$

Our Poisson distribution is then instantiated for $\lambda = 11$, allowing us to find the cumulative probabilities associated with each K value. This provides a link to the chosen quantile q, with the cumulative probability defined as

$$\Pr[K \leq k] = \sum_{i=0}^{k} \frac{e^{-\lambda} \cdot \lambda^j}{j!} \geq q$$

The result of this computation for $k \in \{1, 2, ..., 29\}$ can be seen in Table 3 (Appendix). For us to arrive within the 99.99% quantile, we must have $k = 25$. That is, with a probability of 99.99% there will be at most 25 win events. Using Equation (4), the lower bound of our premium can be calculated to be

$$m \geq 0.425$$

which also satisfies Equation (3). This means that the insurance premium per unit stake must be at least 0.425. Therefore, assuming that there are $N = 10^8$ estimated number of total plays, I would charge a £0.43 premium per unit stake as this corresponds to an extremely low probability ($\leq 0.01\%$) of bankruptcy. The code detailing the calculations above can be viewed in the `Q4.py` file.

Question 5:

The expected value $a(i, j)$ for $7 \leq i \leq 19$ and $7 \leq j \leq 9$ can be computed recursively by starting with the known payouts for $i = 20$ and $7 \leq r \leq 9$ provided in Table 2. For each draw, one of two possible events can occur: either one of your chosen numbers is drawn (i.e. you have a match) or none of your numbers is drawn (i.e. you have missed). The number of remaining draws is $20 - i$, the number of remaining possible matches is $10 - j$ and the number of missed selections is $(80 - 10) - (i - j)$. The probability of matching on the next draw is then

$$\Pr[\text{match}] = \frac{10 - j}{80 - i}$$

and the probability of missing is

$$\Pr[\text{miss}] = \frac{70 - (i - j)}{80 - i}$$

If we match, the next expected value jumps to $a(i+1, j+1)$, whereas if we miss, the next expected value becomes $a(i+1, j)$. For a given i, j, the recursive form of the expected value then given by the following equation:

$$a(i, j) = a(i + 1, j + 1) \cdot \Pr[\text{match}] + a(i + 1, j) \cdot \Pr[\text{miss}]$$

We begin the recursive loop using the boundary condition

$$a(20, j) = P(j)$$

where $P(j)$ corresponds to the payoff for j matches as seen in Table 2. The complete table can be seen in Table 4 (Appendix), with the relevant computations outlined in the `Q5.py` file.

Question 6:

For a fair Keno machine, the probability distribution for selecting each number is uniform. Given that the numbers are drawn without repeats, it follows that for any number $i \in \{1, 2, ..., 80\}$, the marginal probability of drawing i is

$$p = \frac{20}{80} = 0.25$$

If a Keno machine had some bias present during selection, it follows that the expected frequency for each of the biased numbers would noticeably differ from the actual frequencies observed. To quantify this, we apply the Pearson residual statistic. For the expected frequency E_i and observed frequency O_i of a selected number $i \in \{1, 2, ..., 80\}$, its Pearson residual is given by

$$r_i = \frac{O_i - E_i}{\sqrt{E_i}} \tag{6}$$

This measures how far the observed frequency deviates from its expected amount in units of standard deviation. Intuitively, if the machine is biased, any biased

number will produce a larger positive residual than any unbiased number. As residuals over 3 standard deviations indicate a significant deviation, this can be used as a threshold to filter the residuals so that the overly selected numbers can be easily identified. Note that, as the question states that we already know that some numbers have a biased selection probability, there is no need to implement a Pearson's chi-squared test to check if any numbers are biased.

We implement the following statistical test to justify this.

1. Run the faulty Keno machine D times and record each set of selected numbers in a list l.
2. Find the frequency O_i of each number $i \in \{1, 2, ..., 80\}$ using l.
3. Compute the expected frequency for each number
$$E_i = D \cdot p = 0.25D \quad \forall\, i \in \{1, 2, ..., 80\}.$$
4. Compute the residuals for each i via Equation (6).
5. Filter the resulting list using a threshold value $T = 3$ such that for sufficiently large r_i:
$$r_i > T.$$
6. The i values remaining post-filtering are overly selected by the faulty Keno machine.

The validity of this procedure is verified by the example implementation provided in `Q6.py`. Here, we model a Keno machine which has a 0.05% increased probability of selecting the numbers 5, 14, 39 and 71. By running the Keno machine 100,000 times and then applying the statistical test, we correctly identify the overly selected numbers as 5, 14, 39 and 71, demonstrating that the test works as expected for machines which select some numbers with a slightly higher probability.

Appendix

k	$\Pr[K \leq k]$
1	0.0002
2	0.0010
3	0.0041
4	0.0130
5	0.0328
6	0.0700
7	0.1295
8	0.2129
9	0.3170
10	0.4338
11	0.5529
12	0.6643
13	0.7604
14	0.8375
15	0.8951
16	0.9355
17	0.9622
18	0.9789
19	0.9887
20	0.9942
21	0.9971
22	0.9986
23	0.9994
24	0.9997
25	0.9999
26	0.9999
27	1.0000
28	1.0000
29	1.0000

Table 3: Cumulative probabilities $\Pr[K \leq k]$ for $K \sim \text{Poisson}(11)$.

Draws (i)	**Matches (j)**	$a(i,j)$
7	7	13802.05
7	8	82294.52
7	9	465753.42
8	7	11295.47
8	8	72288.73
8	9	437500.00
9	7	9061.18
9	8	62684.10
9	9	408450.70
10	7	7099.42
10	8	53527.95
10	9	378571.43
11	7	5408.10
11	8	44872.12
11	9	347826.09
12	7	3982.40
12	8	36773.49
12	9	316176.47
13	7	2814.15
13	8	29294.44
13	9	283582.09
14	7	1891.22
14	8	22503.50
14	9	250000.00
15	7	1196.70
15	8	16475.96
15	9	215384.62
16	7	708.09
16	8	11294.64
16	9	179687.50
17	7	396.16
17	8	7050.69
17	9	142857.14
18	7	223.74
18	8	3844.53
18	9	104838.71
19	7	144.26
19	8	1786.89
19	9	65573.77

Table 4: Expected values $a(i,j)$ for $7 \leq i \leq 19$ and $7 \leq j \leq 9$.