

Report 9

Maximilian Fernaldy - C2TB1702

Part 1 - Poisson Distribution

Introduction to the Poisson Distribution

The Poisson distribution expresses the probability of a given number of events happening in a certain amount of time, if they occur with a known, constant mean rate and the events are independent of each other (meaning, an event happening does not change the probability of more events happening after it).

The Poisson distribution can be expressed as an equation of probability of n events happening in a fixed amount of time:

$$P(X_t = k) = \frac{e^{-\lambda t} (\lambda t)^k}{k!}$$

With $P(X_t = k)$ as probability of k events happening in time t and λ as average number of events that has happened in time t .

Something to note is that while λ can be any positive real number, k is only defined for positive integers. Let's do an example problem to understand why this is.

An example

Suppose a restaurant is visited by 13.5 customers per day on average. The restaurant is underground, so people can't see inside and consequently, how many customers are inside the store doesn't affect the probability of new customers visiting it.

We can then denote $\lambda = 13.5$ as the average amount of customers per day and $t_0 = 1d = 24h$ as the reference time interval.

An investor is interested in purchasing part ownership of the restaurant for a single trial week, but he wants to know if the restaurant will turn a profit. To get any return on investment (ROI), the investor needs 90 or more customers in that week. He has a statistician calculate the probability of this happening.

Now we can get a sense of why X_t and k has to be integers. It doesn't make sense to try and calculate the probability of, say, 66.66 customers visiting the restaurant in a week. Events happen in integers: either they happen or they don't, 0 or 1.

The statistician uses the Poisson distribution to do the task assigned to them. They note that the investor wanted to know the probability of 90 or more customers visiting the restaurant ($k \geq 90$) in $t_q = 7$ days, which means an easy way to calculate this probability is to instead calculate the inverse probability, then subtract it from unity:

$$P(X_t \geq 90) = 1 - P(0 \leq X_t < 90)$$

However, the statistician quickly realizes that plugging in these values one by one like this:

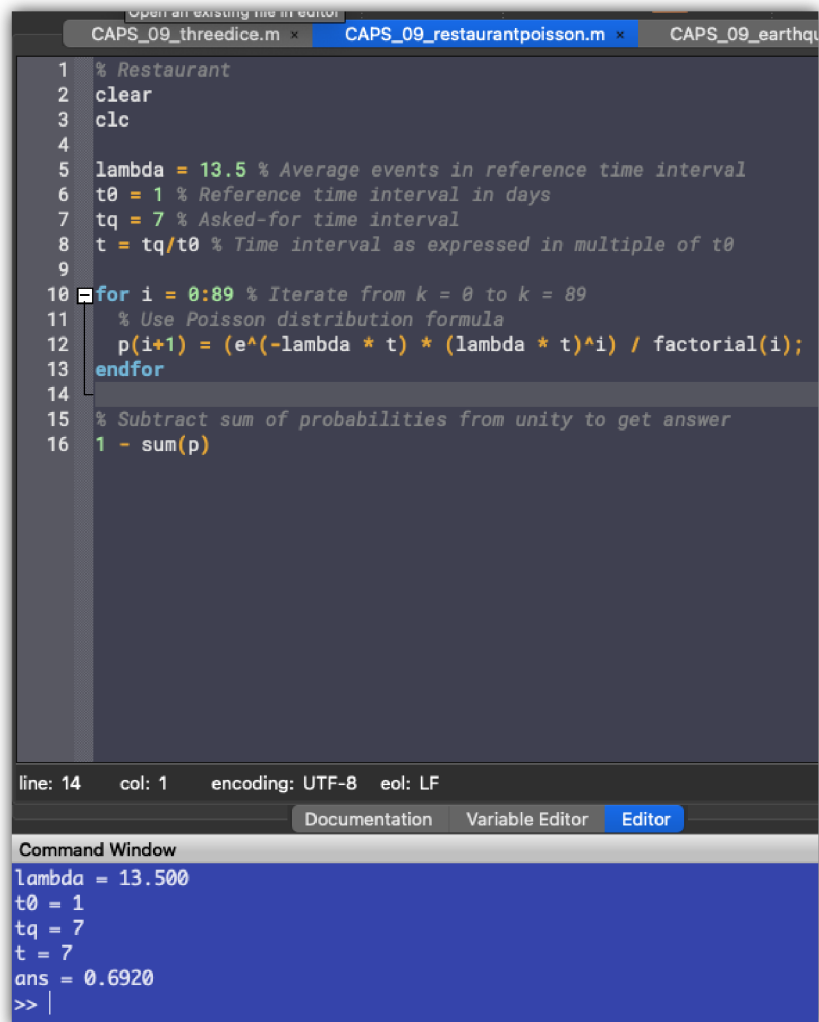
$$P(0 \leq X_t < 90) = P(X_t = 0) + P(X_t = 1) + \dots + P(X_t = 89)$$

will take too long and it's too much work for how much they're getting paid. Instead, the statistician starts up Octave and writes this code to compute the probability:

```
lambda = 13.5 % Average events in reference time interval
t0 = 1 % Reference time interval in days
tq = 7 % Asked-for time interval
t = tq/t0 % Time interval as expressed in multiple of t0

for i = 0:89 % Iterate from k = 0 to k = 89
    % Use Poisson distribution formula
    p(i+1) = (e^(-lambda * t) * (lambda * t)^i) / factorial(i);
endfor

% Subtract sum of probabilities from unity to get answer
1 - sum(p)
```



```
1 % Restaurant
2 clear
3 clc
4
5 lambda = 13.5 % Average events in reference time interval
6 t0 = 1 % Reference time interval in days
7 tq = 7 % Asked-for time interval
8 t = tq/t0 % Time interval as expressed in multiple of t0
9
10 for i = 0:89 % Iterate from k = 0 to k = 89
11     % Use Poisson distribution formula
12     p(i+1) = (e^(-lambda * t) * (lambda * t)^i) / factorial(i);
13 endfor
14
15 % Subtract sum of probabilities from unity to get answer
16 1 - sum(p)
```

line: 14 col: 1 encoding: UTF-8 eol: LF

Documentation Variable Editor Editor

Command Window

```
lambda = 13.500
t0 = 1
tq = 7
t = 7
ans = 0.6920
>> |
```

Figure 1 - Output of restaurantpoisson.m

Apparently, the probability of 90 or more customers visiting the restaurant is 69.2%. The investor is happy with this percentage, and goes through with his decision.

Solving Exercise 9.1

Exercise 9.1

- A, B, C and D are the last 4 digits from your student number (see Exercise 4.1)
- In an area of a country, it is known that earthquakes occur $0.7 \cdot (A+1)$ times in $B+1$ days in an average sense since the dawn of the history
- However, there were $10+C+D$ earthquakes in the last four weeks
- Calculate the probability of $10+C+D$ and more earthquakes occurring in four consecutive weeks

10

Parameters

My student number is C2TB1702.

1. Earthquakes occur $0.7 \times (1 + 1) = 1.4$ times in $7 + 1 = 8$ days on average.
2. There were $10 + 0 + 2 = 12$ earthquakes in the last four weeks.

Applying Poisson Distribution

From parameter (1), $\lambda = 1.4$ and the reference time period is $t_0 = 8$ days. From parameter (2), $k = 12$ and $t_q = 28$ days.

We want to know $P(X_t \geq 12)$, which we can calculate by subtracting its inverse from unity.

$$P(X_t \geq 12) = 1 - P(0 \leq X_t \leq 11)$$

To calculate the inverse probability:

$$P(0 \leq X_t \leq 11) = \sum_{i=0}^{11} P(X_t = i) = \sum_{i=0}^{11} \frac{e^{-\lambda t} (\lambda t)^i}{i!}$$

Because λ is defined for the reference time interval t_0 , to continue to use λ , we want to linearly adjust t so that it is a ratio of t_q and t_0 . We can do this by dividing t_q by t_0 , which will leave use with the appropriate number for λt . In octave code:

```

lambda = 1.4; % Average number of earthquakes in 8 days
t0 = 8; % Reference time period
tq = 28;
t = tq/t0; % Expressed as multiple of t0
k = 12; % Limiter for events

for i = 0:k - 1 % Iterate from 0 to 11
    prob(i+1) = (lambda * t)^i * e^(-lambda * t) / factorial(i);
endfor

```

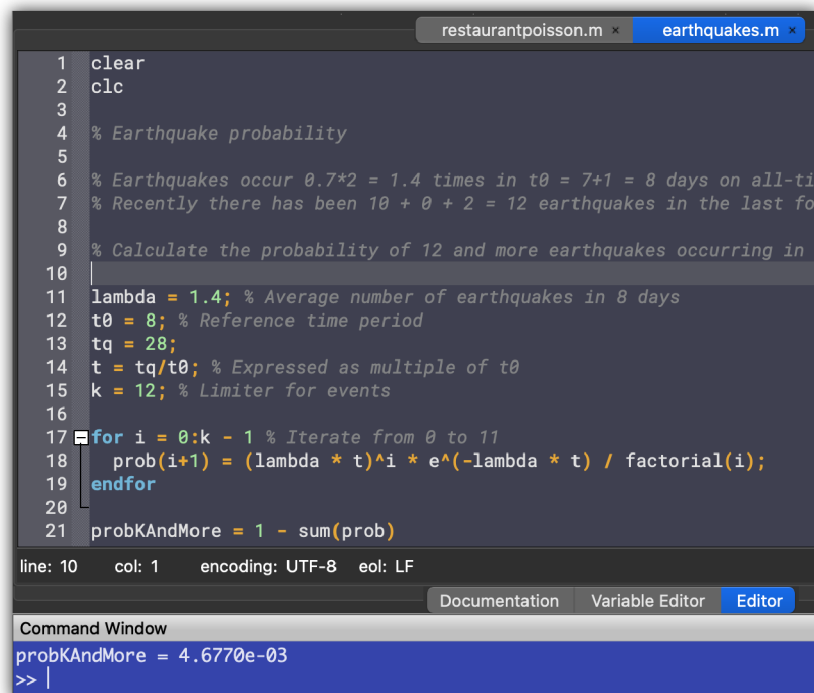
Then simply subtracting it from unity, we get the probability for 12 or more earthquakes happening ($P(X_t \geq 12)$):

```

probKAndMore = 1 - sum(prob)

```

which outputs



The screenshot shows a MATLAB script editor with two tabs: 'restaurantpoisson.m' and 'earthquakes.m'. The 'earthquakes.m' script contains the following code:

```

1 clear
2 clc
3
4 % Earthquake probability
5
6 % Earthquakes occur 0.7*2 = 1.4 times in t0 = 7+1 = 8 days on all-ti
7 % Recently there has been 10 + 0 + 2 = 12 earthquakes in the last fo
8
9 % Calculate the probability of 12 and more earthquakes occurring in
10
11 lambda = 1.4; % Average number of earthquakes in 8 days
12 t0 = 8; % Reference time period
13 tq = 28;
14 t = tq/t0; % Expressed as multiple of t0
15 k = 12; % Limiter for events
16
17 for i = 0:k - 1 % Iterate from 0 to 11
18     prob(i+1) = (lambda * t)^i * e^(-lambda * t) / factorial(i);
19 endfor
20
21 probKAndMore = 1 - sum(prob)

```

The Command Window at the bottom shows the output of the script:

```

probKAndMore = 4.6770e-03
>>

```

Figure 2 - Output of earthquakes.m

This means the probability of 12 earthquakes or more happening is *very* small under normal circumstances - smaller than 0.4%. This suggests that something else might be at play, something out of the ordinary that caused the sudden jump in earthquake frequency.

Part 2 - Three Dice

Exercise 9.2

- Let's roll 3 six-faced dices.
- Calculate the probability of the sum of the three dices is equal to the sum of the last 4 digits from your student number.
- What are the number combinations ?

11

Parameters

The sum of the last 4 digits of my student number is $1 + 7 + 0 + 2 = 10$.

Applying basic probability theory

When rolling three dice, the number of possible sums is actually only 16. The lowest possible sum is $1 + 1 + 1 = 3$ and the highest possible sum is $6 + 6 + 6 = 18$. Therefore there are only 16 possible different sums. However, they have different odds of happening. For example, to get a sum of 3, there is only one possible combination of the three dice, that being all of them showing 1. On the other hand, there are many different combinations that result in a sum of 7. This means the probability of getting one sum is not equal to the others.

This is why when calculating the probability of getting a certain sum, we can't just say it's $\frac{1}{16}$. We need to consider all the possible events and take note of the combinations that result in that sum.

Applying this to our problem, there are $6^3 = 216$ possible combinations of the three dice, and the possible events that result in a sum of 10 are:

1. $1 + 3 + 6$ with unique numbers, so $3! = 6$ events
2. $1 + 4 + 5$ with unique numbers, so $3! = 6$ events
3. $2 + 2 + 6$ with a number appearing twice, so $3!/2! = 3$ events
4. $2 + 3 + 5$ with unique numbers, so $3! = 6$ events
5. $2 + 4 + 4$ with a number appearing twice, so $3!/2! = 3$ events
6. $3 + 3 + 4$ with a number appearing twice, so $3!/2! = 3$ events

Adding them all up, we have $6 \times 3 + 3 \times 3 = 27$ events. The probability of getting a sum of 10 is:

$$\frac{27}{216} = \frac{1}{8} = 12.5\%$$

Using Octave to get the probability

Counting the different probable events one by one is impractical and prone to mistakes. Instead, we can use the help of computers to do the counting for us.

It's quite apparent that a quick and easy way to do this is utilizing nested `for` loops to repeatedly check the sum of the three dice and add the combination to an array if it equals 10.

```
wantedSum = 10;

combinations = []; % Initialize empty array
for i = 1:6 % Iterate through all possible combinations
    for j = 1:6
        for k = 1:6
            if i + j + k == wantedSum % If the sum matches
                newComb = sort([i,j,k]'); % sort the combination
                % put the combination inside the array
                combinations = [combinations; newComb];
            endif
        endfor
    endfor
endfor
```

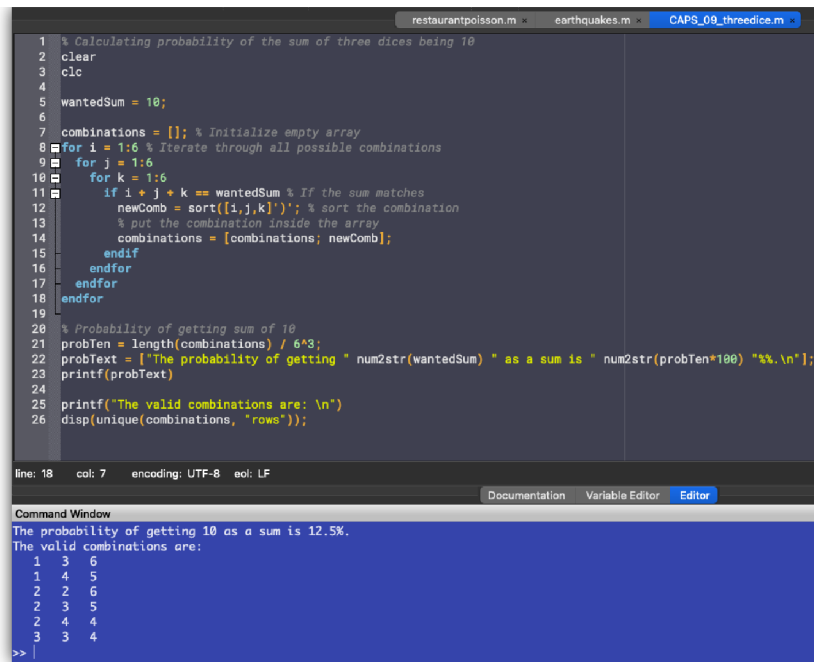
Before adding into the array, we sort the combination so we can take out the duplicates later. To compute the probability, we divide the total number of occurrences `length(combinations)` by the total number of possible events, $6^3 = 216$.

```
% Probability of getting sum of 10
probTen = length(combinations) / 6^3;
probText = ["The probability of getting " num2str(wantedSum) " as a sum is " num2str(probTen*100) "%%."];
printf(probText)
```

To display the combinations that result in a sum of 10, we can use the `unique()` function. This function searches for duplicates, deletes them and sorts the remaining entries in ascending order. The documentation for `unique()` can be found [here](#).

```
printf("The valid combinations are: \n")
disp(unique(combinations, "rows")); % Display unique combinations
```

As we don't need to show differently ordered, but otherwise identical combinations, this function is perfect for our purpose. The output of this script is:



The image shows a MATLAB script in the editor window and its output in the command window. The script, named `threedice.m`, calculates the probability of the sum of three dice being 10. It uses nested loops to generate all possible combinations of three dice (1-6) and checks if their sum equals the desired sum (10). The probability is calculated as the number of valid combinations divided by the total number of possible combinations (6^3 = 216). The output shows the probability is 12.5% and lists the 10 valid combinations.

```
1 % Calculating probability of the sum of three dices being 10
2 clear
3 clc
4
5 wantedSum = 10;
6
7 combinations = []; % Initialize empty array
8 for i = 1:6 % Iterate through all possible combinations
9     for j = 1:6
10        for k = 1:6
11            if i + j + k == wantedSum % If the sum matches
12                newComb = sort([i,j,k]); % sort the combination
13                % put the combination inside the array
14                combinations = [combinations; newComb];
15            endif
16        endfor
17    endfor
18 endfor
19
20 % Probability of getting sum of 10
21 probTen = length(combinations) / 6^3;
22 probText = ['The probability of getting ' num2str(wantedSum) ' as a sum is ' num2str(probTen*100) '%.\n'];
23 printf(probText)
24
25 printf('The valid combinations are: \n')
26 disp(unique(combinations, 'rows'));
```

line: 18 col: 7 encoding: UTF-8 eol: LF

Documentation Variable Editor Editor

Command Window

The probability of getting 10 as a sum is 12.5%.

The valid combinations are:

```
1 3 6
1 4 5
2 2 6
2 3 5
2 4 4
3 3 4
```

>> |

Figure 3 - Output of threedice.m

Comparing this to our earlier manual computation, we get the same result.