Question 1

Min, Max, and Exponential

Posted on March 1, 2013 by Jonathan Mattingly | Comments Off

Let X_1 and X_2 be random variables and let $M = \max(X_1, X_2)$ and $N = \min(X_1, X_2)$.

1. Argue that the event $\{M \le x\}$ is the same as the event $\{X_1 \le x, X_2 \le x\}$ and similarly that t the event $\{N > x\}$ is the same as the event $\{X_1 > x, X_2 > x\}$.
2. Now assume that the X_1 and X_2 are independent and distributed with c.d.f. $F_1(x)$ and $F_2(x)$ respectively . Find the c.d.f. of M and the c.d.f. of N using the proceeding observation.
3. Now assume that X_1 and X_2 are independently and exponentially distributed with parameters λ_1 and λ_2 respectively. Show that N is distributed exponentially and identify the parameter in the exponential distribution of N.
4. The route to a certain remote island contains 4 bridges. If the time to collapse of each bridge is exponential distributed with mean 20 years and is independent of the other bridges, what is the distribution of the time until the road is impassable because one of the bridges has collapsed.

1) if $M \le x$ then $\max(X_1, X_2) \le x$

let if $M = X_1$ then $X_2 \le X_1 \le x$
if $M = X_2$ then $X_1 \le X_2 \le x$

$\Rightarrow \{X_1 \le x, X_2 \le x\}$

if $N > x$ then $\min(X_1, X_2) > x$

if $N = X_1$ then $X_2 > X_1 > x$
if $N = X_2$ then $X_1 > X_2 > x$

$\Rightarrow \{X_1 > x, X_2 > x\}$

2) X_1, X_2 independent with cdf $F_1(x), F_2(x)$

$$\mathbb{P}(X_1 \leq x) = F_1(x)$$

cdf of $M = F_m(x) = \mathbb{P}(M \leq x) = \mathbb{P}(X_1 \leq x, X_2 \leq x)$

$$= \mathbb{P}(X_1 \leq x)\mathbb{P}(X_2 \leq x)$$

$$\therefore F_m(x) = F_1(x) \cdot F_2(x)$$

cdf of $N = F_N(x) = \mathbb{P}(N \leq x) = 1 - \mathbb{P}(N > x)$

$$= 1 - \mathbb{P}(\min(X_1, X_2) > x) = 1 - \mathbb{P}(X_1 > x)\mathbb{P}(X_2 > x)$$

$$= 1 - (1 - \mathbb{P}(X_1 \leq x))(1 - \mathbb{P}(X_2 \leq x))$$

$$= 1 - (1 - F_1(x))(1 - F_2(x))$$

3) if $X_1 \sim \exp(\lambda_1)$ then $F_1(x) = 1 - e^{-\lambda_1 x}$

$X_2 \sim \exp(\lambda_2) \implies F_2(x) = 1 - e^{-\lambda_2 x}$

then $F_n(x) = 1 - (1 - 1 + e^{-\lambda_1 x})(1 - 1 + e^{-\lambda_2 x})$

$$= 1 - e^{-(\lambda_1 + \lambda_2)x}$$

$\therefore N \sim \exp(\lambda_1 + \lambda_2)$ (λ parameter $= \lambda_1 + \lambda_2$)

4) 4 bridges

let T_i be the time to collapse of bridge i

$T_i \sim \exp\left(\frac{1}{20}\right)$ $\qquad f(t) = \frac{1}{20} e^{-\frac{1}{20}t}$

$\lambda = \frac{1}{\mu}$ $\Rightarrow F_{T_i}(t) = 1 - e^{-\frac{1}{20}t}$

let X be distribution of time until road is impassable

$X \sim \min(T_1, T_2, T_3, T_4)$ $\qquad \because$ one of the bridges has collapsed

$$F_X(t) = \mathbb{P}(X \le t)$$

$$= 1 - \mathbb{P}(X > t) = 1 - \mathbb{P}(\min(T_i) > t) = 1 - \mathbb{P}(T_1 > t)\,\mathbb{P}(T_2 > t)\,\mathbb{P}(T_3 > t)\,\mathbb{P}(T_4 > t)$$

$$\Rightarrow 1 - e^{-\frac{1}{20} \times 4t} = 1 - e^{-\frac{1}{5}t}$$

time until bridge collapse $X \sim \exp\left(\frac{1}{5} \text{ years}\right)$

Question 2

> ⓘ **Exercise 8.1: Sitting at Tables**
>
> A banquet table has $2n$ seats. n people are invited to the banquet and each person brings a friend. Each of the $2n$ people are seated randomly at the table so all arrangements are equally likely. Let N be the number of people sitting next to the friend they came with. What is $\mathbf{E}N$?

n people invited to banquet

$N =$ # of people who, from people invited, get seated next to their friend

$$N = \sum_{i=1}^{n} \mathbb{1}_{A_i} \qquad A_i = \{i\text{th person sat next to their friend}\}$$

$$\mathbb{E}(N) = \sum_{i=1}^{n} \mathbb{E}(\mathbb{1}_{A_i})$$

$$\mathbb{E}(\mathbb{1}_{A_i}) = 1 \cdot \mathbb{P}(\mathbb{1}_{A_i} = 1) + 0 \cdot \mathbb{P}(\mathbb{1}_{A_i} = 0) = \mathbb{P}(A_i)$$

$\mathbb{P}(A_i) = \mathbb{P}$ ith person sits next to their friend.

2n-1 seats to choose from

of these 2 seat person next to friend

$$\therefore \mathbb{P}(A_i) = \frac{2}{2n-1}$$

$$\therefore \mathbb{E}(N) = \sum_{i=1}^{n} \mathbb{P}(A_i) = \frac{2n}{2n-1}$$

Question 3

Consider an office that is mailing out k letters to n district individuals. Unbeknownst to the person sending out the letter, each letter was personalized to the independent recipient. They just stuffed the letters in to a random envelope with no regard as to whom the letter was addressed.

Let N be the number of letters that we mailed to the right person. If we define A_i to be the event that the ith letter was mailed to the right person, then

$$N = \sum_{i=1}^{k} \mathbf{1}_{A_i}$$

Clearly the events A_i and A_j are not independent! For example, if person i gets person j letter than person j can not get the right letter either. However for any A_j, $\mathbf{P}(A_j) = \frac{1}{k}$ so

$$\mathbf{E}N = \sum_{i=1}^{k} \mathbf{E}\mathbf{1}_{A_i} = \sum_{i=1}^{k} \mathbf{P}(A_i) = k\frac{1}{k} = 1$$

Thus, on average one person gets the right letter and this is independent of the number of letters being sent out!

ⓘ **Exercise 8.2: Number of Correctly Stuffed Letters Looks Poisson**

Write python code to stimulate stuffing the letters in random envelopes as described in the "Stuffing Envelops" section. Consider the number of letters n equal to 5, 10, 100, 1000, 10000.

Let $\hat{N}_n(k)$ be the frequency of you observe from your simulation of seeing k letters in the correct envelope when there are n letters. Let $X \sim \mathrm{Poisson}(1)$ and define $p_k = \mathbf{P}(X = k)$. (Here $\mathrm{Poisson}(1)$ is the Poisson distribution with parameter 1.)

1. Compare p_k to $\hat{N}_n(k)$ for $k = 0, 1, 2, 3, 4, 5, 6$ and $n = 5, 10, 100, 1000, 10000$. Make a table.
2. The *Total Variation* distance of two distributions is the sum of the absolute difference of their probabilities. In this setting the total variation between the empirical distribution and the $\mathrm{Poisson}(1)$ distribution is given by

$$\mathrm{distance}_{TV}(\hat{N}_n, X) = \sum_{k=0}^{\infty} |\hat{N}_n(k) - p_k|$$

   Plot this for different values of n. Clearly you can't code an infinite sum. Just truncate at a big enough value of k where p_k and $\hat{N}_n(k)$ are very small.

Part 1

In [149]:
```python
#here I am just creating a function

import numpy as np

def create_shuffled_array_pair(n):
    numbers = np.array(list(range(1, n+1)))
    shuffled_numbers = np.random.permutation(numbers)
    return numbers, shuffled_numbers


norm, shuffled = create_shuffled_array_pair(10)

print(norm)
print(shuffled)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[ 7  2  9  3  5  4  8 10  1  6]
```

In [150]:
```python
#table that has the obsereved frequency at each value of n

def observed_frequency_for_n(n, frequency_table, num_trials=100):

    for i in range(num_trials):
        #create an observation
        norm, shuffled = create_shuffled_array_pair(n)

        # print(len(norm))
        # count how many elements are in the same position
        numItemsInSamePosition = 0
        for i in range(len(norm)):
            if norm[i] == shuffled[i]:
                numItemsInSamePosition += 1
        if (numItemsInSamePosition <= 6):
            frequency_table[n][numItemsInSamePosition] += 1
```

In [151]:

```python
n_to_consider = [5, 10, 100, 1000, 10000]


nk_table = {}
# initial population of the frequency table with zeros
for n in n_to_consider:
    nk_table[n] = [0,0,0,0,0,0,0]


for n in n_to_consider:
    observed_frequency_for_n(n, nk_table)

print(nk_table)

print(np.sum(nk_table[5]))
```

```
{5: [34, 41, 10, 11, 0, 4, 0], 10: [35, 36, 23, 4, 2, 0, 0], 100: [39
, 36, 16, 5, 4, 0, 0], 1000: [41, 36, 18, 5, 0, 0, 0], 10000: [39, 34
, 18, 6, 3, 0, 0]}
100
```

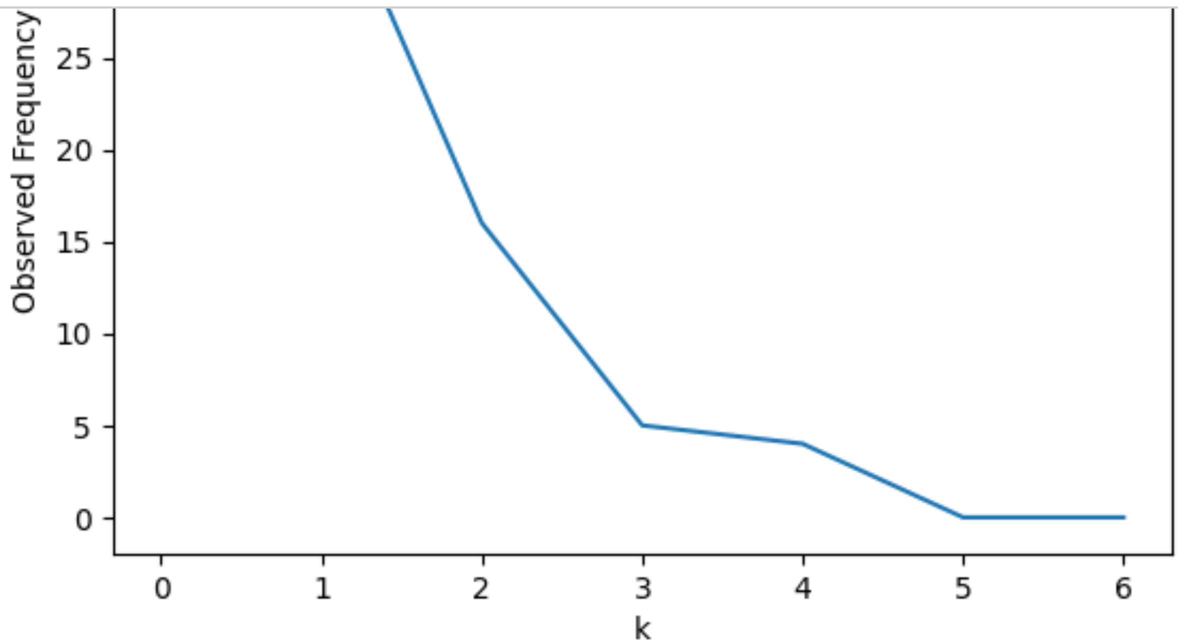
In [152]:

```
# this is me being silly

import matplotlib.pyplot as plt


# Plotting n = 5
plt.plot(nk_table[5])
plt.title('Observed Frequency for n = 5')
plt.xlabel('k')
plt.ylabel('Observed Frequency')
plt.show()

for n in n_to_consider:
    plt.plot(nk_table[n])
    plt.title(f'Observed Frequency for n = {n}')
    plt.xlabel('k')
    plt.ylabel('Observed Frequency')
    plt.show()
```



In [153]:

```
from tabulate import tabulate
from scipy.stats import poisson

def probForFreq(freq, n):
    return freq/n

def poiss(k, lambda_val=1):
    return poisson.pmf(k, lambda_val)

num_trials = 100

# Define the table data
table_data = [
    ["K = 0", probForFreq(nk_table[5][0], num_trials), probForFreq(nk_
    ["K = 1", probForFreq(nk_table[5][1], num_trials), probForFreq(nk_
    ["K = 2", probForFreq(nk_table[5][2], num_trials), probForFreq(nk_
    ["K = 3", probForFreq(nk_table[5][3], num_trials), probForFreq(nk_
    ["K = 4", probForFreq(nk_table[5][4], num_trials), probForFreq(nk_
    ["K = 5", probForFreq(nk_table[5][5], num_trials), probForFreq(nk_
    ["K = 6", probForFreq(nk_table[5][6], num_trials), probForFreq(nk_
]

# Define the table headers
headers = ["k value", "n = 5", "n = 10", "n = 100", "n = 1000", "n = 1

# Print the table
print(tabulate(table_data, headers=headers))
```

```
k value      n = 5    n = 10    n = 100    n = 1000    n = 10000
Poisson
---------  -------  --------  ---------  ----------  -----------  ---
--------
K = 0         0.34      0.35       0.39        0.41         0.39  0.3
67879
K = 1         0.41      0.36       0.36        0.36         0.34  0.3
67879
K = 2         0.1       0.23       0.16        0.18         0.18  0.1
8394
K = 3         0.11      0.04       0.05        0.05         0.06  0.0
613132
K = 4         0         0.02       0.04        0            0.03  0.0
153283
K = 5         0.04      0          0           0            0     0.0
0306566
K = 6         0         0          0           0            0     0.0
00510944
```

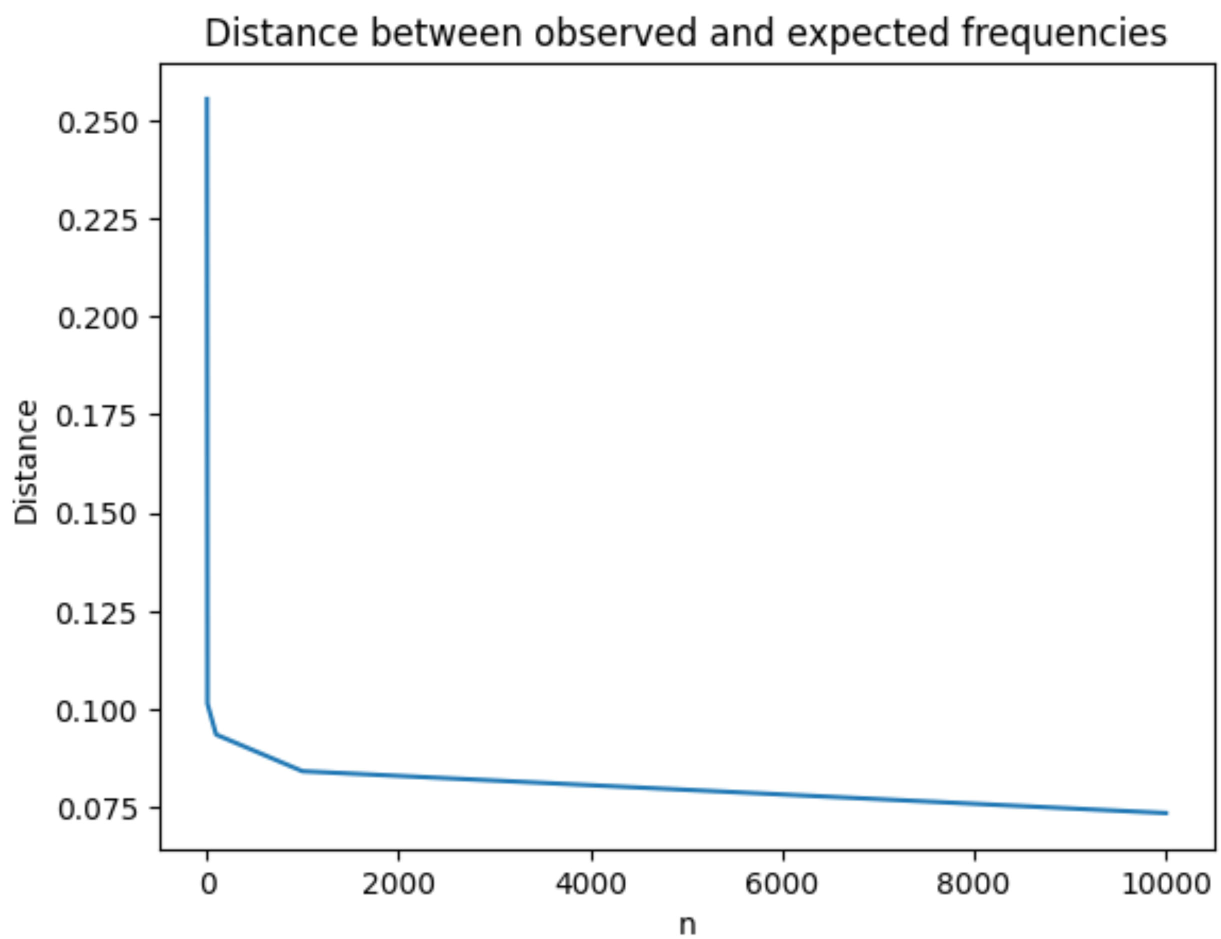
In [154]:

```
def distance(observed, expected):
    return np.sum(np.abs(observed - expected))

error = {}

for n in n_to_consider:
    currentError = 0
    for k in range(7):
        currentError += (distance(probForFreq(nk_table[n][k],num_trial
        # print(distance(probForFreq(nk_table[n][k],num_trials), poiss
    error[n] = currentError

plt.plot([5,10,100,1000,10000], [error[5],error[10],error[100],error[1
plt.title('Distance between observed and expected frequencies')
plt.xlabel('n')
plt.ylabel('Distance')
plt.show()
```



In [ ]: