

Queueing Theory: Empirical distributions.

EBB074A05

Nicky D. van Foreest

2020:12:16

1 General info

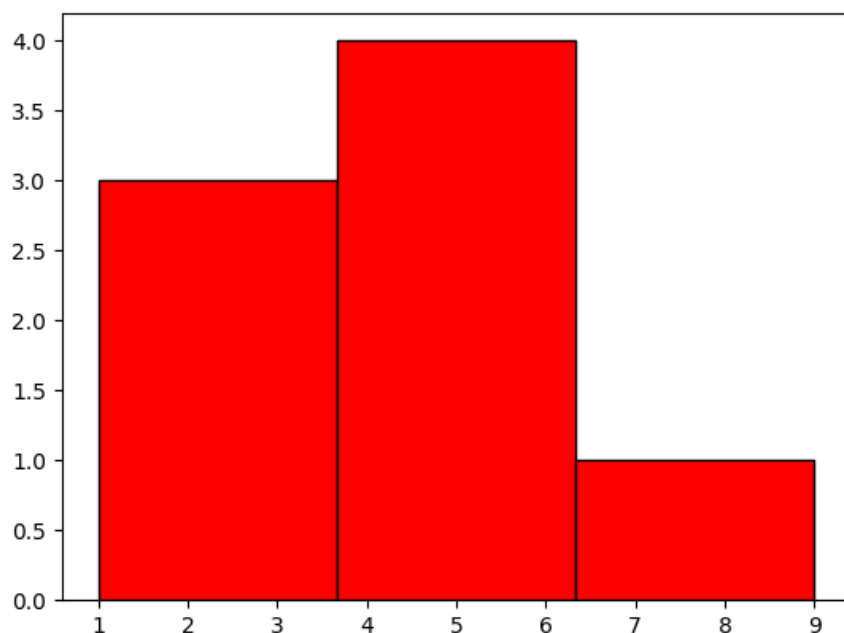
This file contains the code and the results that go with this youtube movie: <https://youtu.be/aKfv908uWqM>

2 Empirical distribution, how to make and plot

We want to know the fraction of periods the queue length is longer than some value q , say. For this we will make the empirical distribution of the queue lengths.

2.1 Plotting a PDF/histogram

```
1 import matplotlib.pyplot as plt
2
3 x = [2, 5, 2, 1, 9, 5, 5, 5]
4
5 plt.clf()
6 plt.hist(x, bins=3, facecolor='red', edgecolor='black', linewidth=1)
7 plt.savefig('emp0.png')
8 'emp0.png'
```



2.1.1 DONE Change the number of bins from 3 to 7.

you can remove the bins argument altogether.

2.2 First naive idea

Given a set of measurements x_1, \dots, x_n , the empirical CDF is defined as

$$F(x) = \sum_{i=1}^n I_{x_i \leq x} / n$$

This is a clean mathematical definition, but as is often the case with mathematical definitions, you should stay clear from using it to *compute* the CDF: the numerical performance is absolutely terrible.

```

1 x = [2, 5, 2, 1, 9, 5, 5, 5]
2
3 def F(y):
4     tot = 0
5     for xi in x:
6         tot += xi <= y
7
8     return tot/len(x)
9
10 print(F(5.5))

```

0.875

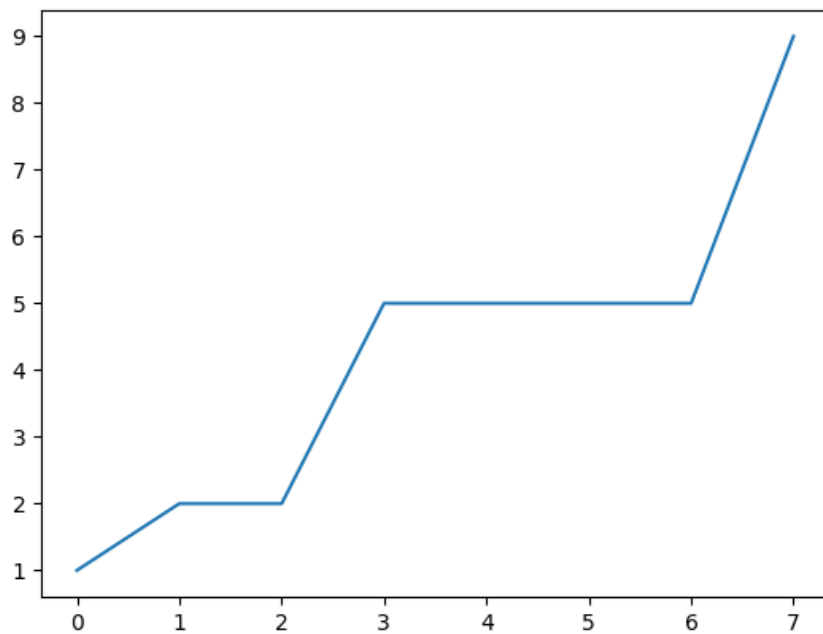
Ex 2.1. Explain in your own words why this way to compute the empirical CDF is a not so smart (i.e., pretty dumb) idea.

2.3 A better idea

```
1 print(sorted(x))
```

```
[1, 2, 2, 5, 5, 5, 5, 9]
```

```
1 plt.clf()
2 plt.plot(sorted(x))
3 plt.savefig("emp00.png")
4 "emp00.png"
```



2.4 Yet better idea

```
1 def cdf_better(x):
2     x = sorted(x)
3     n = len(x)
4     y = range(1, n + 1)
5     y = [z / n for z in y] # normalize
6     return x, y
7
8
9 x = [2, 5, 2, 1, 8, 5, 5]
10 x, F = cdf_better(x)
11 print(F)
```

```
[0.14285714285714285, 0.2857142857142857, 0.42857142857142855, 0.5714285714285714, 0.7142857142857143, 0.8571428571428571, 0.9285714285714286]
```

2.4.1 DONE Explain

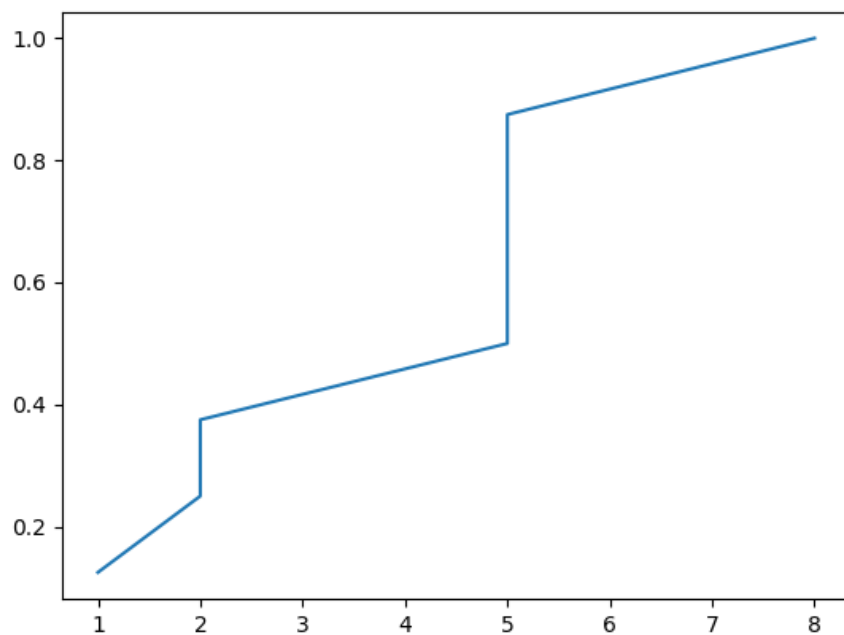
Why the $\sim n = \text{len}(x)$

2.4.2 TODO Explain

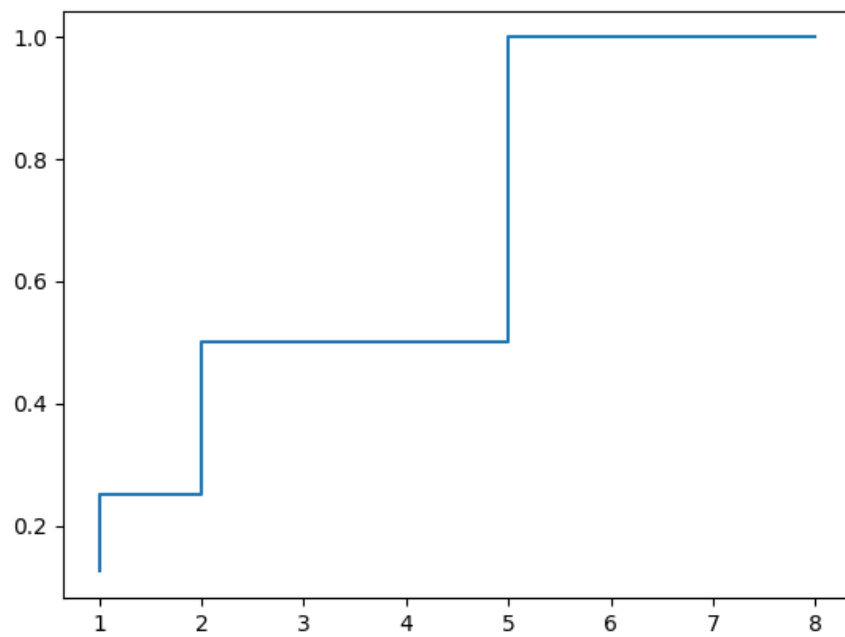
You should know that for loops in R and python are quite slow. We use this in the list comprehension in the line in which we `#normalize`. For larger amounts of data it is better to use numpy. This we do below.

2.5 Plot the cdf

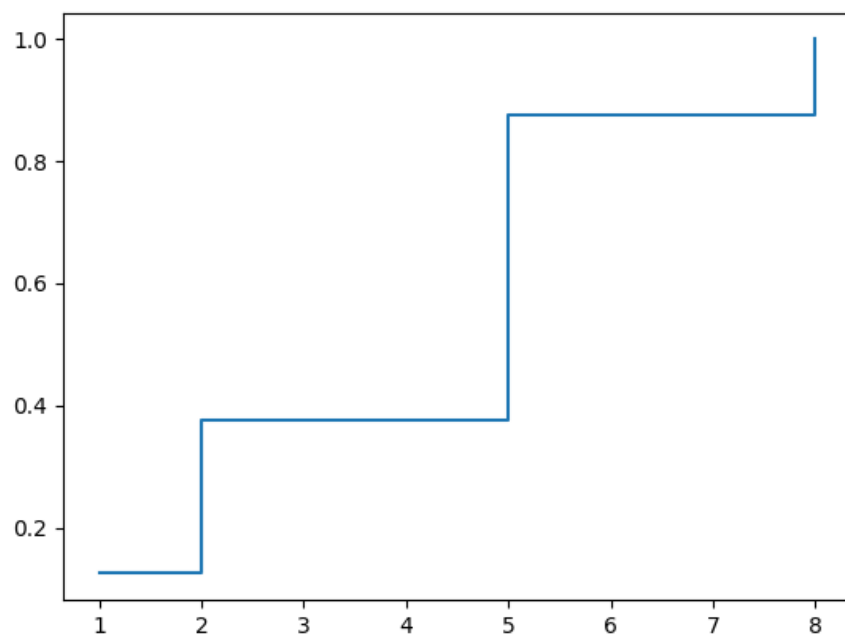
```
1 x = [2, 5, 2, 1, 8, 5, 5, 5]
2 x, F = cdf_better(x)
3
4 plt.clf()
5 plt.plot(x, F)
6 plt.savefig(fname)
7 fname
```



```
1 plt.clf()
2 plt.step(x, F)
3 plt.savefig(fname)
4 fname
```



```
1 plt.clf()
2 plt.plot(x, F, drawstyle="steps-post")
3 plt.savefig(fname)
4 fname
```



2.6 Faster with numpy

```
1 import numpy as np
2
3 def cdf(x):
4     y = np.arange(1, len(x) + 1) / len(x)
5     x = np.sort(x)
6     return x, y
7
8 x = [2, 5, 2, 1, 8, 5, 5]
9 x, F = cdf(x)
10 print(F)
```

```
[0.14285714 0.28571429 0.42857143 0.57142857 0.71428571 0.85714286
 1.          ]
```

2.7 Remove duplicate values

Finally, we can make the computation of the cdf significantly faster with using the following numpy functions.

```
1 unique, count = np.unique(np.sort(x), return_counts=True)
2 print(unique, count)
```

```
[1 2 5 8] [1 2 3 1]
```

```
1 print(count.cumsum()/7)
```

```
[0.14285714 0.42857143 0.85714286 1.          ]
```

```
1 def cdf_fastest(x):
2     # remove multiple occurrences of the same value
3     unique, count = np.unique(np.sort(x), return_counts=True)
4     x = unique
5     y = count.cumsum() / count.sum()
6     return x, y
7
8 x = [2, 5, 2, 1, 8, 5, 5]
9 x, F = cdf_fastest(x)
10 print(F)
```

```
[0.14285714 0.42857143 0.85714286 1.          ]
```

Ex 2.2. What does `unique` do? (Look it up on the web.) What happens if you forget to sort the input `x`?

Ex 2.3. Find some data set on the web—take any data source you like—and compute the empirical CDF. Include a graph of the ecdf, and comment on your findings. You can compare your results with the `EDCF` function of the `statsmodels` module (search on google), if you are interested.

Ex 2.4. Explain what the Kolmogorov-Smirnov test has to do with ecdf's; search on the web for this. Keep your discussion short, but to the point.