Queueing Theory: Psychiatrists doing intakes

EBB074A05

Nicky D. van Foreest

2020:12:17

1 General info

This file contains the code and the results that go with this youtube movie: `https://youtu.be/bCU3oP6r-00`.

There are some exercises for you to make for your assignments.

1.1 TODO Set theme and font size

Set the theme and font size so that it is easier to read on youbute

2 Base situation

5 psychiatrists do intakes. See my queueing book for further background.

2.1 Load standard modules

```
import numpy as np
import matplotlib.pylab as plt
from matplotlib import style

style.use('ggplot')

np.random.seed(3)
```

2.2 Simulate queue length

```
def computeQ(a, c, Q0=0):  # initial queue length is 0
    N = len(a)
    Q = np.empty(N)  # make a list to store the values of  Q
    Q[0] = Q0
    for n in range(1, N):
        d = min(Q[n - 1], c[n])
        Q[n] = Q[n - 1] + a[n] - d
    return Q
```

Ex 2.1. Compute the queue lenghts when

$$a = [1,2,5,6,8,3,7,3], \qquad c = [2,2,0,5,4,4,3,2].$$

2.3 Arrivals

We start with run length 10 for demo purpose.

```
a = np.random.poisson(11.8, 10)
print(a)
```

```
[12  9  7 13 14  9  9 11 12 10]
```

2.4 Service capacity

```
def unbalanced(a):
    p = np.empty([5, len(a)])
    p[0, :] = 1.0 * np.ones_like(a)
    p[1, :] = 1.0 * np.ones_like(a)
    p[2, :] = 1.0 * np.ones_like(a)
    p[3, :] = 3.0 * np.ones_like(a)
    p[4, :] = 9.0 * np.ones_like(a)
    return p

p = unbalanced(a)
print(p)
```

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]
 [9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]]
```

2.5 Include holidays

```
def spread_holidays(p):
    for j in range(len(a)):
        psych = j % 5
        p[psych, j] = 0

spread_holidays(p)
print(p)
```

```
[[0. 1. 1. 1. 1. 0. 1. 1. 1. 1.]
 [1. 0. 1. 1. 1. 1. 0. 1. 1. 1.]
 [1. 1. 0. 1. 1. 1. 1. 0. 1. 1.]
 [3. 3. 3. 0. 3. 3. 3. 3. 0. 3.]
 [9. 9. 9. 9. 0. 9. 9. 9. 9. 0.]]
```

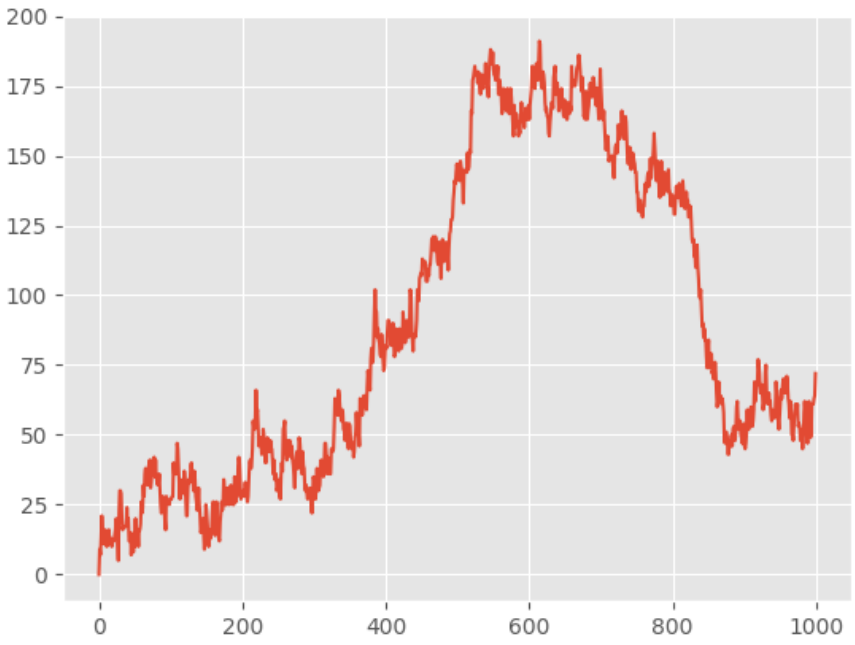
2.6 Total weekly service capacity

```
s = np.sum(p, axis=0)
print(s)
```

```
[14. 14. 14. 12.  6. 14. 14. 14. 12.  6.]
```

2.7 Simulate the queue length process

```
np.random.seed(3)

a = np.random.poisson(11.8, 1000)
p = unbalanced(a)
spread_holidays(p)
s = np.sum(p, axis=0)

Q1 = computeQ(a, s)

plt.clf()
plt.plot(Q1)
plt.savefig("psych1.png")
"psych1.png"
```



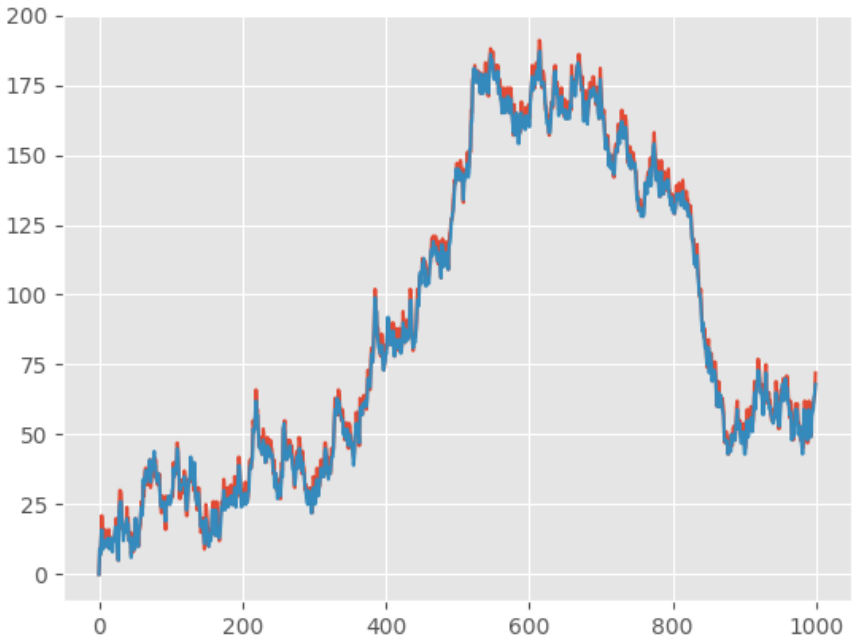
3 Evaluation of better (?) plans

3.1 Balance the capacity more evenly over the psychiatrists

I set the seed to enforce a start with the same arrival pattern.

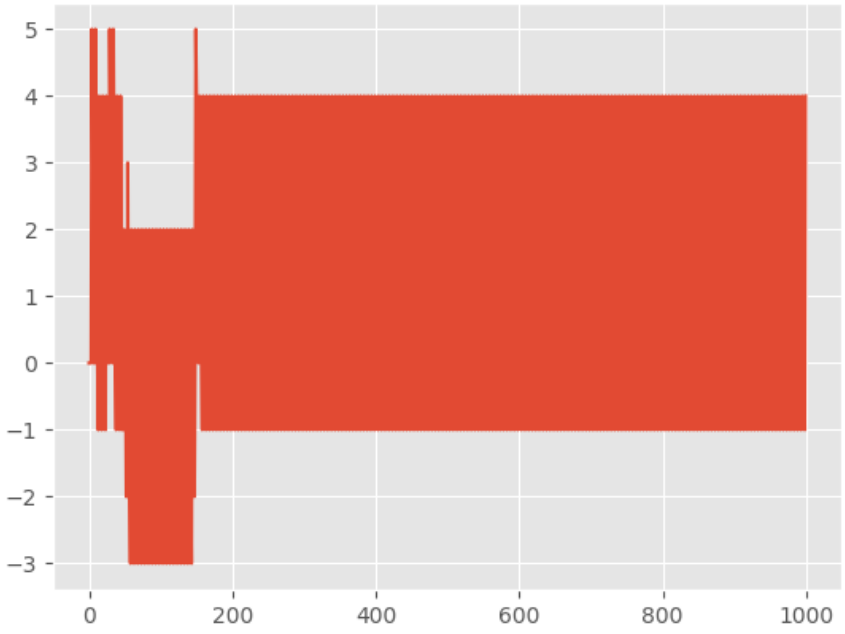
```
def balanced(a):
    p = np.empty([5, len(a)])
    p[0, :] = 2.0 * np.ones_like(a)
    p[1, :] = 2.0 * np.ones_like(a)
    p[2, :] = 3.0 * np.ones_like(a)
```

```
    p[3, :] = 4.0 * np.ones_like(a)
    p[4, :] = 4.0 * np.ones_like(a)
    return p

np.random.seed(3)
a = np.random.poisson(11.8, 1000)


p = balanced(a)
spread_holidays(p)
s = np.sum(p, axis=0)
Q2 = computeQ(a, s)

plt.plot(Q2)
plt.savefig("psych2.png")
"psych2.png"
```



What is the effect?

```
plt.clf()
plt.plot(Q1-Q2)
plt.savefig("psych22.png")
"psych22.png"
```



The effect of balancing capacity is totally uninteresting.

3.2 Synchronize holidays

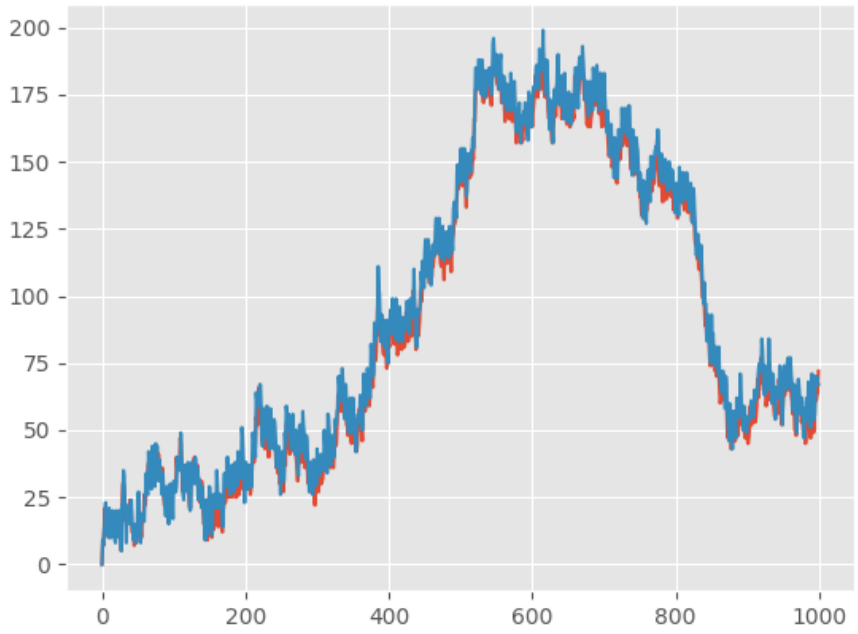
What is the effect of all psychiatrists taking holidays in the same week?

```
a = np.random.poisson(11.8, 10)


def synchronize_holidays(p):
    for j in range(int(len(a) / 5)):
        p[:, 5 * j] = 0

p = unbalanced(a)
synchronize_holidays(p)
print(p)
```

```
[[0. 1. 1. 1. 1. 0. 1. 1. 1. 1.]
 [0. 1. 1. 1. 1. 0. 1. 1. 1. 1.]
 [0. 1. 1. 1. 1. 0. 1. 1. 1. 1.]
 [0. 3. 3. 3. 3. 0. 3. 3. 3. 3.]
 [0. 9. 9. 9. 9. 0. 9. 9. 9. 9.]]
```

```
np.random.seed(3)

a = np.random.poisson(11.8, 1000)
p = unbalanced(a)
spread_holidays(p)
```

```
s = np.sum(p, axis=0)
Q3 = computeQ(a, s)

plt.clf()
plt.plot(Q3)

p = balanced(a)
synchronize_holidays(p)
s = np.sum(p, axis=0)
Q4 = computeQ(a, s)

plt.plot(Q4)
plt.savefig("psych3.png")
"psych3.png"
```



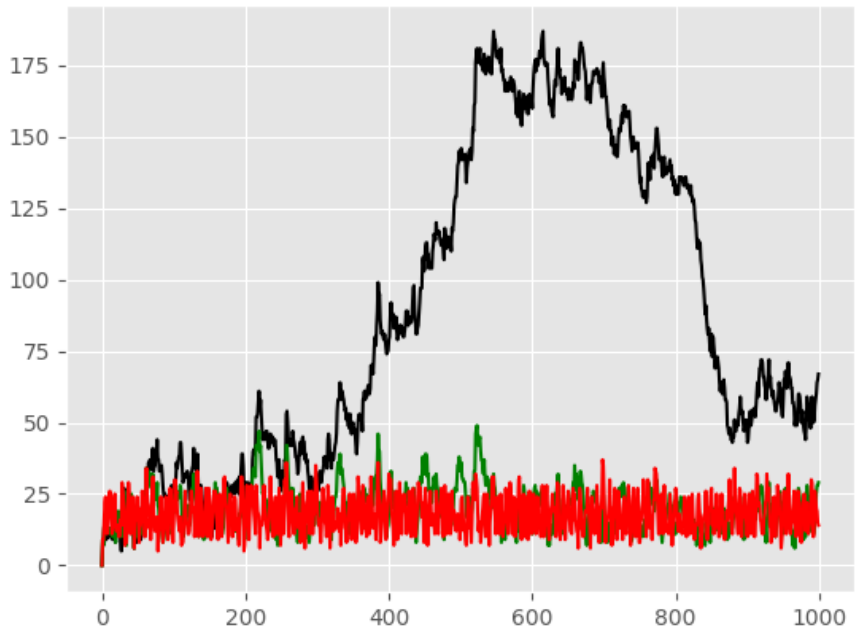
All these proposals will not solve the problem. We need something smarter. For this, we steal an idea from supermarkets: dynamic control.

Ex 3.1. Just to improve your coding skills (and your creativity), formulate another vacation plan. Implement this idea in code, and test its success/failure. Make a graph to show its effect on the dynamics of the queue length. (I don't mind whether your proposal works or not; as long as you 'play' and investigate, all goes.) Include your code (and if you ported all this code to R , then include your R code), and comment on difficult points.

4 Control capacity as a function of queue length

```
lower_thres = 12
upper_thres = 24
```

```
3
4  def computeQExtra(a, c, e, Q0=0):  #  initial queue length is 0
5      N = len(a)
6      Q = [0] * N  # make a list to store the values of  Q
7      Q[0] = Q0
8      for n in range(1, N):
9          if Q[n - 1] < lower_thres:
10             C = c - e
11         elif Q[n-1] >= upper_thres:
12             C = c + e
13         d = min(Q[n-1], C)
14         Q[n] = Q[n-1] + a[n] -d
15     return Q
16
17
18 np.random.seed(3)
19 a = np.random.poisson(11.8, 1000)
20 c = 12
21 Q = computeQ(a, c * np.ones_like(a))
22 Qe1 = computeQExtra(a, c, 1)
23 Qe5 = computeQExtra(a, c, 5)
24
25 plt.clf()
26 plt.plot(Q, label="Q", color='black')
27 plt.plot(Qe1, label="Qe1", color='green')
28 plt.plot(Qe5, label="Qe5", color='red')
29 plt.savefig("psychfinal.png")
30 "psychfinal.png"
```



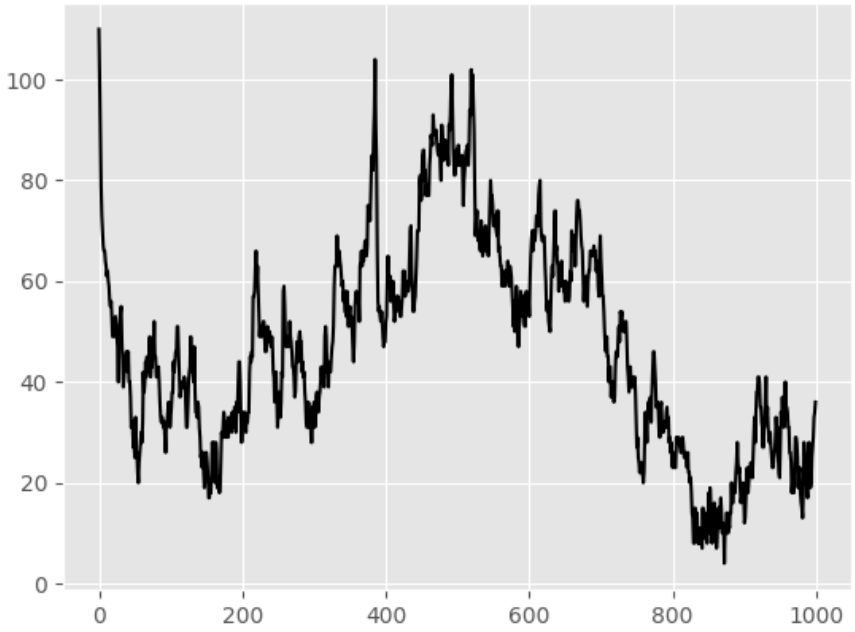
We see, dynamically controlling the service capacity (as a function of queue length) is a much better plan.

Ex 4.1. Use simulation to show that the psychiatrists don't have more work.

Ex 4.2. In the real case the psychiatrists hired an extra person to do intakes when the queue became very long, 100 or higher, and then they hired this person for one month (you may assume that a month consists of 4 weeks). Suppose this person can do 2 intakes a day and works for 4 days a week.

First explain the code below. Then do a number of experiments to see the effect of the duration of the contract (make it longer, or shorter), or the number of intakes per day done by the extra person. In other words, if you were a consultant, what would you advice the psychiatrists?

```
import numpy as np
import matplotlib.pylab as plt
from matplotlib import style

style.use('ggplot')
np.random.seed(3)

extra_capacity = 8  # extra weekly capacity
contract_duration = 4  # weeks


def compute_Q_control(a, c, Q0=0):
    N = len(a)
    Q = np.empty(N)
    Q[0] = Q0
    extra = False
    mark_time = 0
    for n in range(1, N):
        if Q[n - 1] > 100:
            extra = True
            mark_time = n
        if extra and n >= mark_time + contract_duration:
            extra = False
        d = min(Q[n - 1], c[n] + extra * extra_capacity)
        Q[n] = Q[n - 1] + a[n] - d
    return Q


a = np.random.poisson(11.8, 1000)
c = 12
Q = compute_Q_control(a, c * np.ones_like(a), Q0=110)
# print(Q)
plt.clf()
plt.plot(Q, label="Q", color='black')
plt.savefig("psych_extra.png")
"psych_extra.png"
```


100
80
60
40
20
0
0
200
400
600
800
1000