Items that fail – solution

Based on the cumulative percentages given in the table, we can say that if we start week 1 with 100 new parts, the numbers remaining at the start of each week are as follows:

week	**parts**	**no. fail this week**	**proportion that fail this week**
1	100	10	1/10
2	90	15	1/6
3	75	25	1/3
4	50	30	3/5
5	20	20	1
6	0		

We can use the proportions that fail each week to make a transition table.

		Now				
		1	2	3	4	5
Next	1	$\frac{1}{10}$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{3}{5}$	1
	2	$\frac{9}{10}$	0	0	0	0
	3	0	$\frac{5}{6}$	0	0	0
	4	0	0	$\frac{2}{3}$	0	0
	5	0	0	0	$\frac{2}{5}$	0

We turn this into a transition matrix

$$\mathbf{M} = \begin{bmatrix} \frac{1}{10} & \frac{1}{6} & \frac{1}{3} & \frac{3}{5} & 1 \\ \frac{9}{10} & 0 & 0 & 0 & 0 \\ 0 & \frac{5}{6} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{5} & 0 \end{bmatrix}.$$

If we define the initial state as $\mathbf{s}_0 = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 \end{bmatrix}^T$, then the state in week i is $\mathbf{s}_i = \mathbf{M}^i \mathbf{s}_0$.

Say n_i is the number of new parts installed in week i. Then n_i is the first entry in $\mathbf{s}_i$.

We can use our transition matrix to find the steady state, an eigenvector corresponding to an eigenvalue of 1. To do this, I wrote some Python code.

```
from sympy import *
M = S('Matrix([[1/10,1/6,1/3,3/5,1],[9/10,0,0,0,0],[0,5/6,0,0,0],[0,0,2/3,0,0],[0,0,0,2/5,0]])')
eigv = M.eigenvects()
for v in eigv:
if v[0] == 1:
    col_sum = 0
    for i in range(5):
        col_sum += v[2][0][i]
    for i in range(5):
        print((1000*v[2][0][i]/col_sum).evalf())
```

The steady state is

$$\mathbf{s} = \begin{bmatrix} 298.5075 \\ 268.6567 \\ 223.8806 \\ 149.2537 \\ 59.7015 \end{bmatrix}$$

So in steady state under the current policy we expect to fit 299 new machine parts each week, at a cost of £2,541.50.

The proposal was that instead of replacing parts as they fail, we agree to replace all parts at fixed intervals. We'll still have to replace any that fail before the fixed interval is up, so we can use our transition matrix to explore the next few time steps after we replace all the parts. Again, I did this by writing some Python code. Remember that n_i is the first element in $\mathbf{s}_i$.

```
from sympy import *
M = S('Matrix([[1/10,1/6,1/3,3/5,1],[9/10,0,0,0,0],[0,5/6,0,0,0],[0,0,2/3,0,0],[0,0,0,2/5,0]])')
s0 = Matrix([[1000],[0],[0],[0],[0]])
for i in range(11):
s = M**i*s0
pprint(s[0].evalf())
```

This computes the following values.

Week	n_i
0	1000
1	100
2	160
3	281
4	377.1
5	349.86
6	229.801
7	286.0341
8	319.8686
9	312.7202
10	289.6731

If we replace all the parts at the end of the ith week, the cost is £2,500 plus the cost of replacing any parts that have failed prior to that point at £8.50 each. Say C_i is the cost of replacing all parts plus the cost of any that have failed in weeks 1 to $i-1$. Then

$$C_i = 2500 + 8.5 \times \sum_{k=1}^{i-1} n_k.$$

The longer we leave it, the more parts have failed in the earlier weeks. For comparison, we might usefully look at the cost per week, $\frac{C_i}{i}$. All parts fail in week 5, so looking at weeks 1–4 we have

Replace all in week i	C_i	Cost per week
1	$2500 + 850 = 3350$	3350
2	$2500 + 850 + 1360 = 4710$	2355
3	$2500 + 850 + 1360 + 2388.5 = 7098.50$	2366.17
4	$2500 + 850 + 1360 + 2388.5 + 3204.5 = 10303$	2575.75
5	$2500 + 850 + 1360 + 2388.5 + 3204.5 + 2975 = 13278$	2655.6

The current steady state running cost was £2,541.50, so replacing in weeks 1, 4 or 5 would be more expensive than what is currently being done. The cheapest option is to replace all parts every two weeks at a cost of £2,355, thus saving £186.50 per week (7.3% of the running costs). This is traditionally where this problem ends.

Sources

- Sasieni, M., Yaspan, A. and Friedman, L. (1959). Operational Research: Methods and Problems. London: John Wiley & Sons.