Math 42 HW 3

Ben Stewart

April 23, 2022

Collaborators: Evan Coons and Jessie Baker

1

The thing that I found most interesting in chapter 3 of Mooney and Swift is that the proportions of each group evens out to be equal to the eigenvector. This was brought up in class, but I still find it crazy that we can predict exactly what the proportions of one population will be to another as time approaches infinity just by looking at the eigenvalues of the corresponding matrix.

2

I found the use of spreadsheets in recurrence relations very interesting as it is a very applicable to modern technology, and most textbooks I have read have not been as up-to-date with current technologies. The absolute addressing feature in spreadsheets is something that I had not previously thought about when working with recurrence relations, and could see myself making the mistake of not using it have I not read this. I also found the part about systems of recurrence relations interesting and more realistic in real world applications.

3

I found the section discussing the data-ink very interesting, as it really made me think about which parts of a visualization are 100% necessary. For example, the simple bar with 35.9 above it seems like a minimalist representation of a number, but the author goes onto explain how the number 35.9 is represented in 6 different ways. This definitely makes you think about how many unnecessary aspects there are in most graphs that go unnoticed.

4

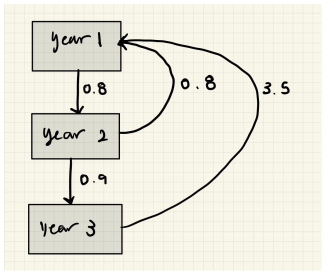
$$x(n) = Rx(n-1) + a$$

$$x(1) = Rx(0) + a$$
$$x(2) = Rx(1) + a = R(Rx(0) + a) + a = R^2x(0) + Ra + a$$
$$x(3) = Rx(2) + a = R(R^2x(0) + Ra + a) = R^3x(0) + R^2a + Ra + a$$

$$x(n) = R^n x(0) + R^{n-1}a + ... + a$$
$$x(n) = R^n x(0) + a(R^{n-1} + R^{n-2} + ... + 1)$$
$$x(n) = R^n x(0) + a\frac{1 - R^n}{1 - R}$$

5

a



b

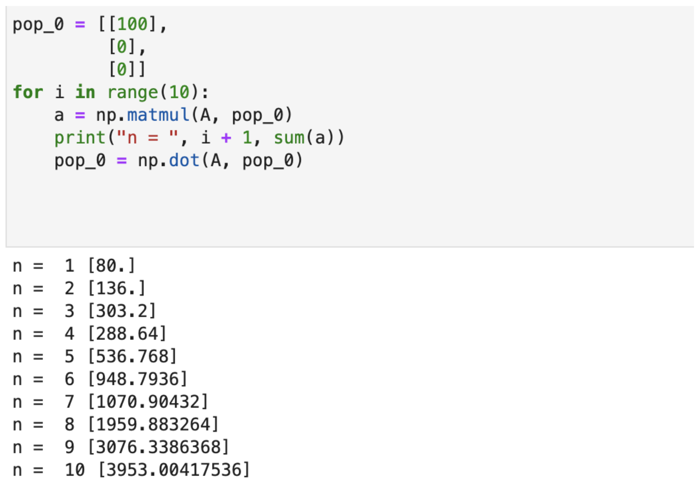
$$\begin{bmatrix} 0 & 0.8 & 3.5 \\ 0.8 & 0 & 0 \\ 0 & 0.9 & 0 \end{bmatrix}$$

c

Eigenvalues: [1.516972, -0.758486+1.04206671i, -0.758486-1.04206671i] with moduli: [1.516972 , 1.28887704, 1.28887704]. Thus, the rate of change of the entire system is dictated by the first eigenvalue to be an increase of 52%.

d

The total female population over ten years is:



```
pop_0 = [[100],
         [0],
         [0]]
for i in range(10):
    a = np.matmul(A, pop_0)
    print("n = ", i + 1, sum(a))
    pop_0 = np.dot(A, pop_0)
```

```
n =  1 [80.]
n =  2 [136.]
n =  3 [303.2]
n =  4 [288.64]
n =  5 [536.768]
n =  6 [948.7936]
n =  7 [1070.90432]
n =  8 [1959.883264]
n =  9 [3076.3386368]
n =  10 [3953.00417536]
```

e

The normalized eigenvector for the dominant eigenvalue is:

$$[0.5434056, 0.28657383, 0.17002057]$$

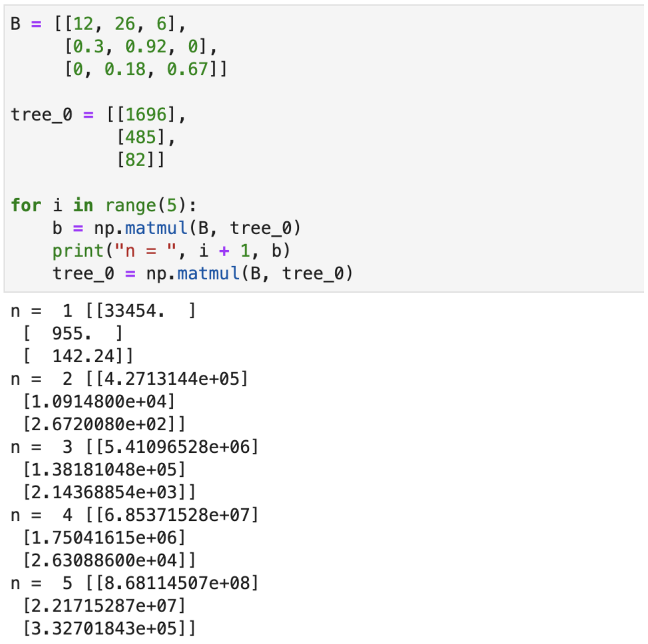
The normalized population distribution is:

$$[0.39752342, 0.37734176, 0.22513482]$$

They are not quite the same yet, but the population distribution will approach the normalized eigenvector as time goes on.

6

This is not possible because the population of trees would quickly grow into millions.



```
B = [[12, 26, 6],
     [0.3, 0.92, 0],
     [0, 0.18, 0.67]]

tree_0 = [[1696],
          [485],
          [82]]

for i in range(5):
    b = np.matmul(B, tree_0)
    print("n = ", i + 1, b)
    tree_0 = np.matmul(B, tree_0)
```

```
n =  1 [[33454.  ]
 [  955.  ]
 [  142.24]]
n =  2 [[4.2713144e+05]
 [1.0914800e+04]
 [2.6720080e+02]]
n =  3 [[5.41096528e+06]
 [1.38181048e+05]
 [2.14368854e+03]]
n =  4 [[6.85371528e+07]
 [1.75041615e+06]
 [2.63088600e+04]]
n =  5 [[8.68114507e+08]
 [2.21715287e+07]
 [3.32701843e+05]]
```

7

7.1

$$\begin{bmatrix} 0.63 & 0.63 & 1.2 & \dots & 1.2 \\ 0.34 & 0 & 0 & \dots & 0 \\ 0 & 0.71 & 0 & \dots & 0 \\ 0 & 0 & 0.71 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0.71 & 0 \end{bmatrix}$$

7.2

The dominant eigenvalue for the matrix is 1.24138336, meaning the population will eventually be growing at 24% . The corresponding eigenvector is: [-9.48524732e-01, -2.59789538e-01, -1.48584698e-01, -8.49819155e-02, -4.86047759e-02, -2.77991409e-02, -1.58995123e-02, -9.09360809e-03, -5.20102167e-03, -2.97468575e-03, -1.70134944e-03, -9.73074185e-04, -5.56542559e-04, -3.18310387e-04, -1.82055263e-04].

When normalized, we get the vector corresponding to the proportion of each age group as time goes to infinity: [6.09911204e-01, 1.67047357e-01, 9.55414964e-02, 5.46442498e-02, 3.12533732e-02, 1.78751349e-02, 1.02235507e-02, 5.84728397e-

03, 3.34431068e-03, 1.91275367e-03, 1.09398527e-03, 6.25696756e-04, 3.57862616e-04, 2.04676868e-04, 1.17063416e-04]

7.3

After tracing the population distribution for ten years, the normalized population distribution is: [6.09895794e-01, 1.67035486e-01, 9.55303318e-02, 5.46353625e-02, 3.12468591e-02, 1.78705907e-02, 1.02204836e-02, 5.84526215e-03, [3.34300122e-03], 1.91191657e-03, 1.09345587e-03, 6.25366902e-04, 3.57659992e-04, 2.04548389e-04, 1.16977569e-04, 6.69040876e-05. This distribution is extremely close to the distribution we saw in part 2, and will only get closer as time goes on.

7.4

$$\begin{bmatrix} 0.6 & 0.6 & 1.15 & \dots & 1.15 \\ 0.32 & 0 & 0 & \dots & 0 \\ 0 & 0.68 & 0 & \dots & 0 \\ 0 & 0 & 0.68 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0.68 & 0 \end{bmatrix}$$

The dominant eigenvalue for this matrix is 1.18285182, meaning the population will grow at 18% per year as time goes on. The corresponding eigenvector for the dominant eigenvalue is [9.49450805e-01, 2.56857412e-01, 1.47662655e-01, 8.48885752e-02, 4.88008981e-02, 2.80547488e-02, 1.61281649e-02, 9.27178871e-03, 5.33018272e-03, 3.06422511e-03, 1.76156729e-03, 1.01269300e-03, 5.82178790e-04, 3.34683999e-04, 1.92403745e-04, 1.10609414e-04]
When we normalize this eigenvector, we get that the distribution for the population will approach [6.11167434e-01, 1.65340726e-01, 9.50513768e-02, 5.46433079e-02, 3.14134440e-02, 1.80590177e-02, 1.03818009e-02, 5.96830851e-03, 3.43107203e-03, 1.97246091e-03, 1.13393191e-03, 6.51876833e-04, 3.74752136e-04, 2.15438188e-04, 1.23851496e-04, 7.11999730e-05]
If we watch the population distribution for ten years, we see that the tenth year population distribution is [6.11145255e-01], [1.65322684e-01], [9.50354888e-02], [5.46364130e-02], [3.14078690e-02], [1.80282721e-02], [1.03441760e-02], [6.05416236e-03], [3.63864385e-03], [1.70199508e-03], [0.00000000e+00], [1.14257045e-03], [5.71285225e-04], [5.71285225e-04], [2.85642612e-04], [1.14257045e-04] which is very close to the predicted distribution and will get closer as time goes on.

8

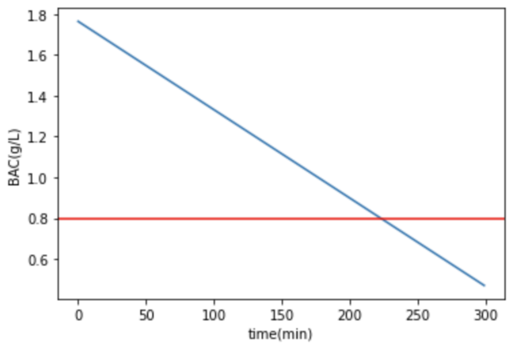
8.1

```
[227]: def bac_male_beers(weight_lbs, beers, time):
           weight_kg = weight_lbs / 2.2046
           fluids = weight_kg * 0.68
           grams = beers * 13.6 - (12 / 60) * time
           grams_per_liter = grams / fluids
           return grams_per_liter
```

```
[228]: bacs = []
       for i in range(300):
           bacs.append(bac_male_beers(150, 6, i))

       x = range(300)
       plt.plot(x, bacs, )
       plt.xlabel("time(min)")
       plt.ylabel("BAC(g/L)")
       plt.axhline(y = 0.8, color = "red")
```

```
[228]: <matplotlib.lines.Line2D at 0x7fb0b87cbf10>
```



It will take 223 minutes for a 150 pound man who frank 6 beers instantaneously to have a BAC less than or equal to 0.8 g/L.

8.2

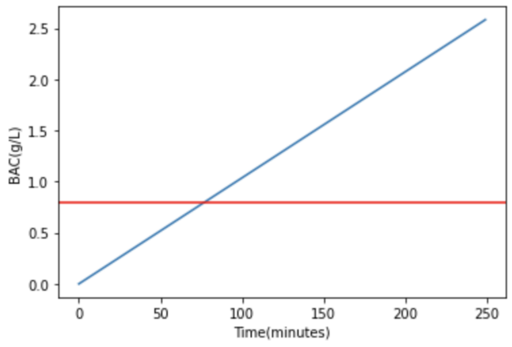
For this section, I modeled a 150 lb male drinking one beer every 20 minutes.

```
[212]: def bac_male_beers(weight_lbs, time):
           weight_kg = weight_lbs / 2.2046
           fluids = weight_kg * 0.68
           grams = (13.6 / 20) * time - (12 / 60) * time
           grams_per_liter = grams / fluids
           return grams_per_liter
```

```
[218]: bacs = []
       for i in range(250):
           bacs.append(bac_male_beers(150, i))
```

```
[225]: plt.plot(range(250), bacs)
       plt.xlabel("Time(minutes)")
       plt.ylabel("BAC(g/L)")
       plt.axhline(y=0.8, color = "red")
```

```
[225]: <matplotlib.lines.Line2D at 0x7fb0b86d1610>
```



8.3

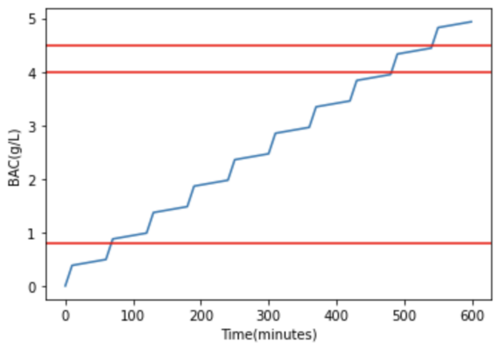
For this section, I modeled a 150 lb male drinking one shot of vodka every hour where the alcohol from the vodka is absorbed into the blood stream in 10 minutes, and drinking one beer every thirty minutes with the alcohol being continuously absorbed into the bloodstream.

```
[270]: bacs = []
       grams = 0
       weight_lbs = 150
       weight_kg = weight_lbs / 2.2046
       fluids = weight_kg * 0.68
       for i in range(600):
           grams += (13.6 / 45)
           grams -= (12 / 60)
           if (i % 60) > 0 and i % 60 <= 10:
               grams += 1.67

           bacs.append(grams / fluids)
```

```
[271]: x = range(600)
       plt.plot(x, bacs)
       plt.xlabel("Time(minutes)")
       plt.ylabel("BAC(g/L)")
       plt.axhline(y = 0.8, color = "red")
       plt.axhline(y = 4, color = "red")
       plt.axhline(y = 4.5, color = "red")
```

```
[271]: <matplotlib.lines.Line2D at 0x7fb0b7d76790>
```



The bottom red line is the legal limit to drive in California, the middle red line is the BAC to put someone in a coma, and the top red line is the BAC for death.