Project3

April 26, 2020

[1]:
```
import numpy as np
from random import random
```

[2]:
```
x = random()
y = random()
print("x = {:.2f}, ".format(x),"y = {:.2f}".format(y))
```

```
x = 0.62,  y = 0.73
```

AA = 0.38 AB = x = 0.62 BA = y = 0.73 BB = 0.27

[3]:
```
M = np.matrix('0.38 0.73 ; 0.62 0.27')
print(M)
```

```
[[0.38 0.73]
 [0.62 0.27]]
```

[11]:
```
v = np.matrix('50; 50');
print("v0 = ")
print(v)
```

```
v0 =
[[50]
 [50]]
```

[12]:
```
print("v0:")
print(v)
for i in range(10):
    print("v{}: ".format(i + 1))
    print(M * v)
    v = M * v
```

```
v0:
[[50]
 [50]]
v1:
[[55.5]
 [44.5]]
```

```
v2:
[[53.575]
 [46.425]]
v3:
[[54.24875]
 [45.75125]]
v4:
[[54.0129375]
 [45.9870625]]
v5:
[[54.09547188]
 [45.90452813]]
v6:
[[54.06658484]
 [45.93341516]]
v7:
[[54.0766953]
 [45.9233047]]
v8:
[[54.07315664]
 [45.92684336]]
v9:
[[54.07439517]
 [45.92560483]]
v10:
[[54.07396169]
 [45.92603831]]
```

Looking at the results, we can expect Vn to be around (54.1, 45.9). The more iterations there are, the closer each element in the vertex gets to the same numbers as the previous iteration.

[16]:
```
v = np.matrix('10; 90');
print("M = ")
print(M)
print("v0 = ")
print(v)


for i in range(10):
    print("v{}: ".format(i + 1))
    print(M * v)
    v = M * v
```

```
M =
[[0.38 0.73]
 [0.62 0.27]]
v0 =
[[10]
```

```
 [90]]
v1:
[[69.5]
 [30.5]]
v2:
[[48.675]
 [51.325]]
v3:
[[55.96375]
 [44.03625]]
v4:
[[53.4126875]
 [46.5873125]]
v5:
[[54.30555937]
 [45.69444062]]
v6:
[[53.99305422]
 [46.00694578]]
v7:
[[54.10243102]
 [45.89756898]]
v8:
[[54.06414914]
 [45.93585086]]
v9:
[[54.0775478]
 [45.9224522]]
v10:
[[54.07285827]
 [45.92714173]]
```

Looking at the results, we can expect Vn to once again be around (54.1, 45.9). The more iterations there are, the closer each element in the vertex gets to the same numbers as the previous iteration. This time, it took a few more iterations to level out perhaps because the distributin of cars have a larger difference.

[18]:
```
v = np.matrix('90; 10');
print("M = ")
print(M)
print("v0 = ")
print(v)


for i in range(10):
    print("v{}: ".format(i + 1))
    print(M * v)
    v = M * v
```

```
M =
[[0.38 0.73]
 [0.62 0.27]]
v0 =
[[90]
 [10]]
v1:
[[41.5]
 [58.5]]
v2:
[[58.475]
 [41.525]]
v3:
[[52.53375]
 [47.46625]]
v4:
[[54.6131875]
 [45.3868125]]
v5:
[[53.88538437]
 [46.11461562]]
v6:
[[54.14011547]
 [45.85988453]]
v7:
[[54.05095959]
 [45.94904041]]
v8:
[[54.08216414]
 [45.91783586]]
v9:
[[54.07124255]
 [45.92875745]]
v10:
[[54.07506511]
 [45.92493489]]
```

Looking at the results, we can expect Vn to once again be around (54.1, 45.9). The more iterations there are, the closer each element in the vertex gets to the same numbers as the previous iteration. This time, The first few iterations are different than the previouis two examples, but they all end up at the same place.

The long term stock of cars in the two cities does not depend on how many cars start out in each city. As seen in the examples, regardless of the starting amount in each city, they all move towards the same points after each iteration. The larger the separation between starting amounts, the more iterations it will take to settle. What would change the long term stock of cars would be to change the matrix.

```
[ ]:
```