Math 42 HW 5

Ben Stewart

May 2022

1

I found the difference between demographic and environmental stochasticity very interesting over time. The web graphs between the two are very different, with the environmental having greater differences between population growth when a random event occurs while the demographic has more variation but with smaller differences in growth. I find the idea of modeling using a uniform distribution between 0 and 1 and having some threshold number to determine whether the event occurs very effective.

2

The Markov Chains with absorbing states are interesting to me because the question shifts from what is the probability of being in state x after a certain number of time steps to how long does it take to reach the absorbing state. I also feel that these models are a little less precise in the sense that we can find the average amount of time steps, but there is a very wide range of possibilities of when the simulation will actually reach the absorbing state. I also think it is cool how we can find the average number of times the model will be in state x given the position it is in.

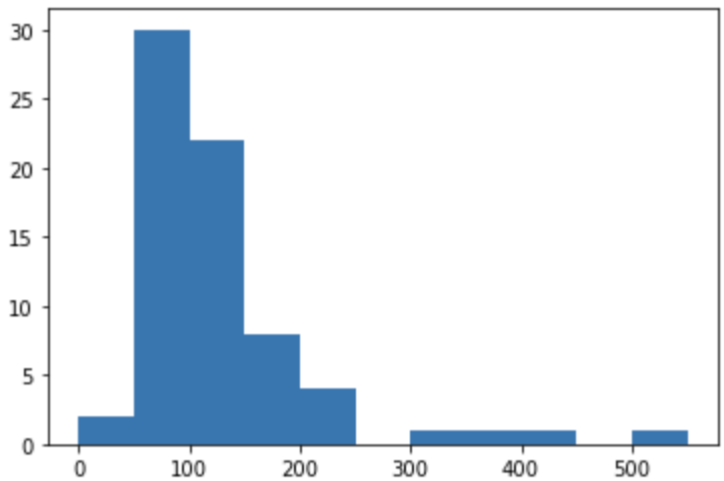
3

The idea that a data point can effectively display more than one piece of information seems to be a very effective in making a model as informative as possible. But, it is very important to do this in a manner that would not make the model confusing or redundant so as to take away from it instead of enhancing it.

4

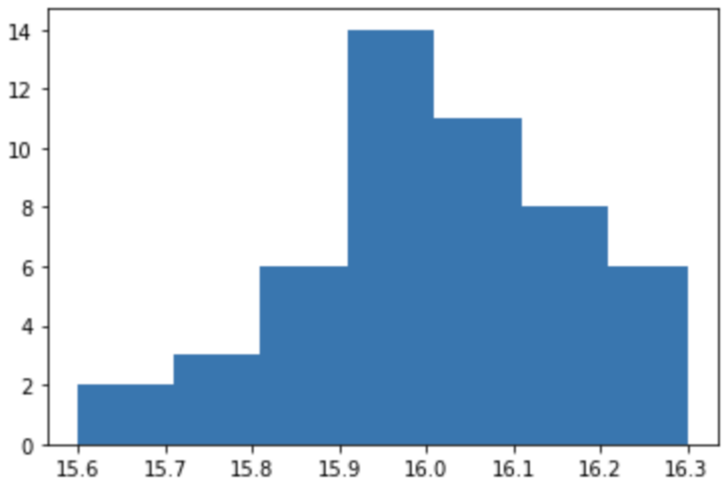
a

Survival Days	Frequency	Relative Frequency
0-50	2	$2/70 = 2.86\%$
51-100	30	$30/70 = 42.86\%$
101-150	22	$22/70 = 31.43\%$
151-200	8	$8/70 = 11.43\%$
201-250	4	$4/70 = 5.71\%$
251-300	0	$0/70 = 0\%$
301-350	1	$1/70 = 1.43\%$
351-400	1	$1/70 = 1.43\%$
401-450	1	$1/70 = 1.43\%$
450-500	0	$0/70 = 0\%$
501-550	1	$1/70 = 1.43\%$



b

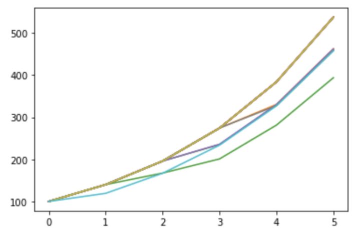
Survival Days	Frequency	Relative Frequency
15.60-15.70	2	$2/70 = 2.86\%$
15.71-15.80	3	$3/70 = 4.29\%$
15.81-15.90	6	$6/70 = 8.57\%$
15.91-16.00	14	$14/70 = 10\%$
16.01-16.1	11	$11/70 = 15.71\%$
16.11-16.20	8	$8/70 = 11.43\%$
16.21-16.30	6	$6/70 = 8.57\%$



5

[113]:
```
def crane_pop_model(cranes, birth_rate, death_rate):
    pop = [cranes]
    for i in range(5):
        cat = np.random.uniform(0, 1)
        if cat < 0.04:
            adjusted_birth_rate = (1 - np.random.uniform(0.38, 0.42)) * birth_rate
            adjusted_death_rate = (1 + np.random.uniform(0.23, 0.23)) * death_rate
        else:
            adjusted_birth_rate = birth_rate
            adjusted_death_rate = death_rate

        cranes =  cranes + adjusted_birth_rate * cranes - death_rate * cranes
        pop.append(cranes)
    return pop
```

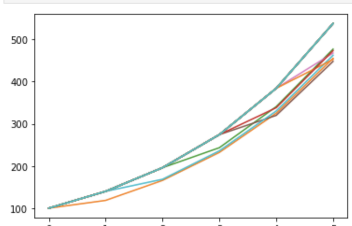
[114]:
```
for i in range(20):
    plt.plot(crane_pop_model(100, 0.5, 0.1))
```



This web graph has much more variability and visible varying populations than the previous model. Even populations that experience catastrophes in the same year do not have the same population growth as both their birth and death rates were most likely effected differently. The lines are more spread out and variable.

[118]:
```
def crane_pop_model_2(cranes, birth_rate, death_rate):
    pop = [cranes]
    for i in range(5):
        cat = np.random.uniform(0, 1)
        if cat < 0.04:
            adjusted_birth_rate = (1 - np.random.uniform(0.30, 0.5)) * birth_rate
            adjusted_death_rate = (1 + np.random.uniform(0.15, 0.30)) * death_rate
        else:
            adjusted_birth_rate = birth_rate
            adjusted_death_rate = death_rate

        cranes =  cranes + adjusted_birth_rate * cranes - death_rate * cranes
        pop.append(cranes)
    return pop
```

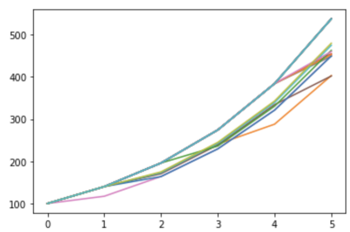
[119]:
```
for i in range(20):
    plt.plot(crane_pop_model_2(100, 0.5, 0.1))
```



In this second model, I made the widened the range of the possible effects of the catastrophe such that the birth rate could decrease by 30% to 50% and the death rate could increase by 15% to 30%. This model shows most populations ending with different population sizes, as all of those effected with catastrophes have a very high chance of having a different effect on their birth and death rates. This graph is even more spread out and variable.

```
[120]: def crane_pop_model_3(cranes, birth_rate, death_rate):
           pop = [cranes]
           for i in range(5):
               cat = np.random.uniform(0, 1)
               if cat < 0.04:
                   adjusted_birth_rate = (1 - np.random.uniform(0.20, 0.70)) * birth_rate
                   adjusted_death_rate = (1 + np.random.uniform(0.10, 0.50)) * death_rate
               else:
                   adjusted_birth_rate = birth_rate
                   adjusted_death_rate = death_rate

               cranes =  cranes + adjusted_birth_rate * cranes - death_rate * cranes
               pop.append(cranes)
           return pop
```

```
[126]: for i in range(20):
           plt.plot(crane_pop_model_2(100, 0.5, 0.1))
```



Finally, I made a model with extremely wide ranges for possible effect on birth and death rate. It appears that the majority of the populations branch out into different ending populations.

6

a

Given that the uniform distribution is

$$f(x) = \begin{cases} \frac{1}{\beta-\alpha} & \text{for } \alpha < x < \beta, \\ 0 & \text{elsewhere} \end{cases}$$

we can prove that the subintervals of (α, β) are equally probable by proving that for some length n such that $n > \beta - \alpha$, the interval $(\alpha, \alpha + n)$ is as probable as the interval $(\beta, \beta + n)$. The probability of the interval $(\alpha, \alpha + n)$ is given by:

$$P = \int_{\alpha}^{\alpha+n} \frac{1}{\beta - \alpha} dx$$

$$P = \left. \frac{x}{\beta - \alpha} \right|_{\alpha}^{\alpha+n}$$

$$P = \frac{n}{\beta - \alpha}$$

The probability of the interval $(\beta, \beta + n)$ is given by:

$$P = \int_{\beta}^{\beta+n} \frac{1}{\beta - \alpha} dx$$

$$P = \frac{x}{\beta - \alpha}\bigg|_{\beta}^{\beta+n}$$

$$P = \frac{n}{\beta - \alpha}$$

Both intervals are of arbitrary length n and have the same probability, thus any intervals on a uniform distribution with the same length will have the same probability.

b

Given that the equation for the mean of a distribution is given by:

$$\mu = \int_a^b xf(x)dx$$

the mean of any uniform distribution is:

$$\mu = \int_\alpha^\beta \frac{x}{\beta - \alpha}dx$$

$$\mu = \frac{x^2}{2(\beta - \alpha)}\bigg|_\alpha^\beta$$

$$\mu = \frac{\beta^2 - \alpha^2}{2(\beta - \alpha)}$$

$$\mu = \frac{\beta - \alpha}{2}$$

Thus, the mean of any uniform distribution is the midpoint.

c

The variance of a relation is given by:

$$V(X) = E(X^2) - (E(X))^2$$

To find the first half of the equation for a uniform distribution:

$$E(X^2) = \int_\alpha^\beta \frac{x^2}{\beta - \alpha}dx$$

$$E(X^2) = \frac{x^3}{3(\beta - \alpha)}\bigg|_\alpha^\beta$$

$$E(X^2) = \frac{\alpha^2 + \alpha\beta + \beta^2}{3}$$

and $E(X)^2$ is given by:

$$E(X)^2 = (\frac{\alpha+\beta}{2})^2$$

$$E(X)^2 = \frac{(\alpha+\beta)^2}{4}$$

Thus,

$$V(X) = \frac{\alpha^2+\alpha\beta+\beta^2}{3} - \frac{(\alpha+\beta)^2}{4}$$

$$V(X) = \frac{4\alpha^2+4\alpha\beta+4\alpha^2-3\alpha^2-6\alpha\beta-3\beta^2}{12}$$

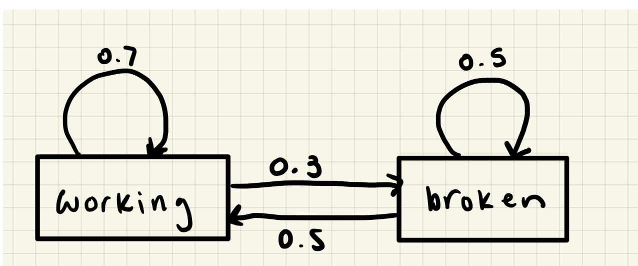
$$V(X) = \frac{\alpha^2-2\alpha\beta+\beta^2}{12}$$

$$V(X) = \frac{(\beta-\alpha)^2}{12}$$

Thus, the variance of a uniform distribution is $V(X) = \frac{(\beta-\alpha)^2}{12}$

7

a



b

The transition matrix is represented by:

$$\begin{bmatrix} 0.7 & 0.5 \\ 0.3 & 0.5 \end{bmatrix}$$

c

```
[128]: mat = np.matrix([[0.7, 0.5], [0.3, 0.5]])

[135]: a = np.matrix([[1], [0]])

[134]: mat @ a

[134]: matrix([[0.7],
                [0.3]])

[136]: mat @ (mat @ a)

[136]: matrix([[0.64],
                [0.36]])

[137]: mat @ (mat @ (mat @ a))

[137]: matrix([[0.628],
                [0.372]])

[140]: np.linalg.matrix_power(mat, 7) @ a

[140]: matrix([[0.6250048],
                [0.3749952]])

[141]: np.linalg.matrix_power(mat, 30) @ a

[141]: matrix([[0.625],
                [0.375]])
```

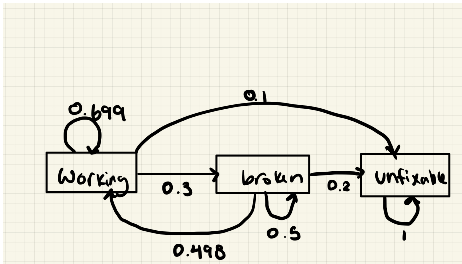
d

```
[43]: np.linalg.eig(mat)

[43]: (array([1. , 0.2]),
       matrix([[ 0.85749293, -0.70710678],
               [ 0.51449576,  0.70710678]]))

[44]: b = np.linalg.eig(mat)[1][:,0]

[46]: b

[46]: matrix([[0.85749293],
              [0.51449576]])

[50]: b * (1 / sum(b))

[50]: matrix([[0.625],
              [0.375]])
```

The long term probability that the washing machine will be working on any given day is 0.625

8

a



b

```
[51]: mat_2 = ([[0.699, 0.498, 0], [0.3, 0.5, 0], [0.01, 0.02, 1]])
```

```
[52]: mat_2
```

```
[52]: [[0.699, 0.498, 0], [0.3, 0.5, 0], [0.01, 0.02, 1]]
```

c

```
[54]: fundamental = np.linalg.inv(np.identity(2) - np.matrix([[0.699, 0.498], [0.3, 0.5]]))
```

```
[55]: fundamental
```

```
[55]: matrix([[454.54545455, 452.72727273],
        [272.72727273, 273.63636364]])
```

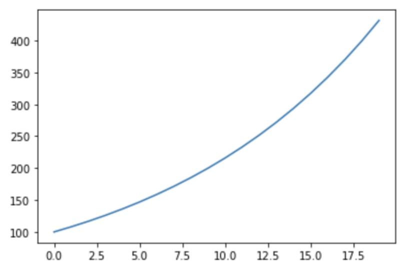
The printer should last around 726 days and it should be working 454 days and under repair for 272 days.

9

9.1

[63]:
```
bobcat_pop = []
bobcats = 100
for i in range(20):
    bobcat_pop.append(bobcats)
    bobcats = bobcats + 0.4 * bobcats - 0.32 * bobcats

plt.plot(bobcat_pop)
```

[63]:
```
[<matplotlib.lines.Line2D at 0x7ffe5fa1a3a0>]
```

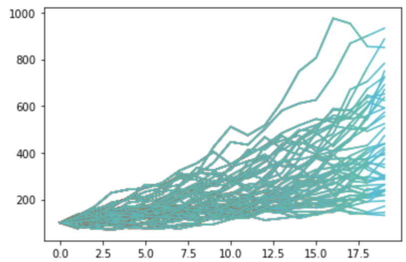


[82]:
```
print(tuple(zip(list(range(0, 20)), bobcat_pop)))
```

```
((0, 100), (1, 108.0), (2, 116.63999999999999), (3, 125.9712), (4, 136.048896), (5, 146.93280768), (6, 158.6874322944), (7, 171.
38242687795199), (8, 185.09302102818813), (9, 199.90046271044318), (10, 215.89249972727868), (11, 233.16389970546098), (12, 251.
8170116818979), (13, 271.9623726164497), (14, 293.7193624257657), (15, 317.216911419827), (16, 342.59426433341315), (17, 370.001
8054800862), (18, 399.60194991849306), (19, 431.57010591197246))
```

9.2

[107]:
```python
bobcat_pop_stoch = []
bobcats = 100
model = pd.DataFrame(dtype=object)
for i in range(50):
    for j in range(20):
        bobcat_pop_stoch.append(bobcats)
        bobcats = np.random.normal(0.68, 0.07, 1) * bobcats + np.random.normal(0.4, 0.1, 1) * bobcats
        plt.plot(bobcat_pop_stoch)
    model[i] = bobcat_pop_stoch
    bobcat_pop_stoch = []
    bobcats = 100
```



[108]:
```python
model
```

[108]:

	0	1	2	3	4	5	
0	100	100	100	100	100	100	
1	[104.55139469722397]	[105.49685932606212]	[114.62897697421499]	[101.36638545438788]	[114.64585182725925]	[134.89505926403945]	[128.791560
2	[127.09849899884588]	[110.29974060424374]	[109.41193032845095]	[119.19151582802016]	[107.41206736885428]	[144.51832313227345]	[167.407808
3	[121.96195844319062]	[108.91158194499089]	[143.94101882711948]	[126.04421438686347]	[128.6730175698828]	[185.63306186476683]	[228.007646
4	[99.4478809142982]	[110.09164143189534]	[157.57392835758384]	[112.49899471251247]	[119.66293655799925]	[201.54356563747365]	[242.36507
5	[95.90817461692782]	[120.42618383986986]	[187.63877887057754]	[132.68219317877197]	[138.21408376885037]	[232.3434363309207]	[247.337609
6	[81.03154808047093]	[105.82562469886345]	[206.9613183546703]	[121.06665327892058]	[167.86378503007256]	[249.1134619266199]	[271.351348
7	[75.08003769289634]	[118.16817174037865]	[240.82093947437164]	[123.4236806063283]	[155.5300317537279]	[261.8861344298655]	[323.06597
8	[89.91246157837811]	[108.54787152430397]	[196.75030419404777]	[116.69287070731795]	[153.4442291222589]	[335.0086043215335]	[354.18583
9	[93.93736500369643]	[142.66800351473796]	[175.78763957380244]	[132.18614904473267]	[171.72745893502284]	[324.40513952708056]	[403.384351
10	[115.07194894737412]	[142.83243365277696]	[172.43180918357422]	[131.56819065550354]	[184.27094521930357]	[350.4461025480794]	[350.551870
11	[130.52855724472514]	[184.5605212909383]	[209.9043241572626]	[145.43380237381692]	[194.81194449924254]	[388.1660799761422]	[394.825071
12	[111.19164719878341]	[188.31825672668538]	[232.4185923091165]	[210.93655722638582]	[210.70832835972283]	[429.84438428465614]	[466.74492
13	[119.15059589668752]	[219.1214245059249]	[228.1018496293749]	[228.76835321287444]	[270.1008718686405]	[483.9141568076942]	[417.6327
14	[128.4791030224937]	[181.29940599468128]	[253.03633966636525]	[264.5726895420789]	[307.57349642366285]	[516.7257254717415]	[390.21823
15	[139.60322369022282]	[163.94554440234265]	[260.6900058162927]	[270.13244066926023]	[293.27505281053226]	[544.1866194845076]	[445.601649
16	[155.2815892205931]	[172.14944780178553]	[238.1546686306561]	[314.20224014213795]	[318.6192256757085]	[531.5564862634407]	[480.60532
17	[146.21013729170716]	[184.65736313655879]	[254.68769679436383]	[284.0193260338409]	[331.3390087094556]	[571.4313425483351]	[626.92191
18	[136.5102119683752]	[214.32910518802913]	[260.1674227610746]	[296.9354451152133]	[377.54459382021935]	[673.2148531448647]	[763.47524
19	[131.58246953060168]	[242.03563252146768]	[264.18033196912177]	[286.0602577277633]	[391.29850421597723]	[725.2459806690438]	[886.48466

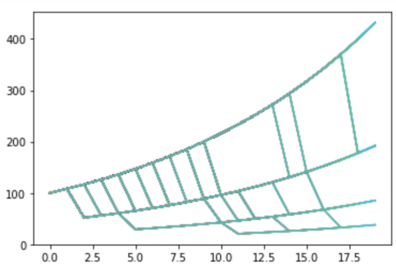
20 rows × 50 columns

43	44	45	46	47	48	49
100	100	100	100	100	100	100
4868432527518]	[96.38921691105426]	[124.850738804827]	[87.21982341976353]	[116.40067566085058]	[123.62500834527782]	[103.04826062869078]
.246467156322]	[119.23441185983572]	[141.90744527612975]	[74.56495063665784]	[130.24503413889914]	[158.39818106068049]	[116.71458505224528]
9603225752997]	[143.99747163784465]	[118.27952730284454]	[71.38045092187087]	[113.89822383922055]	[177.79826146554876]	[134.56443496571683]
0280219938798]	[144.77653077839093]	[130.95013240485537]	[80.78839970654182]	[117.31023000395108]	[199.1438317014211]	[131.0813492334421]
7836732419926]	[134.403624511632]	[137.5586880630362]	[76.65475439371536]	[130.72121928923988]	[263.1047210783666]	[127.3871658298836]
.187577525504]	[159.7049552237546]	[145.529771475497]	[74.79507670370079]	[123.05874753741296]	[254.63049826880427]	[125.7341351882591]
4190641375353]	[166.17956783051775]	[160.6478684227777]	[93.37244554393043]	[157.5769857425634]	[289.9802411841124]	[120.86976702103746]
3882025920577]	[217.19178484859503]	[151.65967793992886]	[99.27120740933276]	[203.1035697213767]	[301.5397784768514]	[125.5937543073928]
7070754096179]	[234.64382765589545]	[148.63367368907126]	[124.64071252873003]	[230.3727880669361]	[357.78198319700846]	[113.47703257572789]
7526100314212]	[203.30699533104212]	[176.69250495225992]	[124.50115764845486]	[253.99181561356187]	[445.8793223692508]	[123.76771331602897]
5626804427158]	[212.96462121151072]	[202.87131535336445]	[170.43172565517796]	[359.6336295778526]	[433.7596791241447]	[132.93617772354105]
3720755772419]	[234.57122488980218]	[208.07145526259887]	[200.02285577012873]	[392.24373927796887]	[502.4141413833463]	[163.87885890106338]
4632696821525]	[247.0369742409556]	[267.5886684730124]	[233.27464718676552]	[342.16203591113333]	[580.5097900389992]	[165.43814319319537]
4673491566384]	[293.6454621707684]	[307.26626503923325]	[274.2545293583424]	[340.9990239004634]	[610.8545132356319]	[163.5638922631161]
.225848228038]	[395.2871067787455]	[376.69886057315296]	[279.1315546680387]	[375.3247476700906]	[625.9707591338293]	[167.774967733425]
.1220267131769]	[468.3651083398785]	[345.94639221651175]	[327.67698295814574]	[373.46128479478716]	[728.8971254728185]	[185.26908418337348]
5023238014054]	[516.5919452787033]	[353.86590021736924]	[323.48500207255336]	[377.18772166270526]	[867.8480134070643]	[217.91826750761214]
.486250682809]	[491.84196349858405]	[410.6414007375538]	[271.9942186802316]	[327.80482058497694]	[899.8047034148831]	[205.08686293924984]
7904492335839]	[523.84340009156]	[436.6241474920583]	[293.1066229993254]	[295.41639547360865]	[932.6843750663701]	[218.11108394333667]

The table is very large so I included the first and last 5 simulations.

9.3

[132]:
```python
bobcat_pop_env = []
bobcats = 100
model2 = pd.DataFrame(dtype=object)
for i in range(50):
    for j in range(20):
        bobcat_pop_env.append(bobcats)
        cat = np.random.uniform(0, 1)
        if cat <= 0.04:
            bobcats = bobcats * 0.38 + 0.1 * bobcats
        else:
            bobcats = bobcats * 0.4 + bobcats * 0.68
        plt.plot(bobcat_pop_env)
    model2[i] = bobcat_pop_env
    bobcat_pop_env = []
    bobcats = 100
```



[133]:

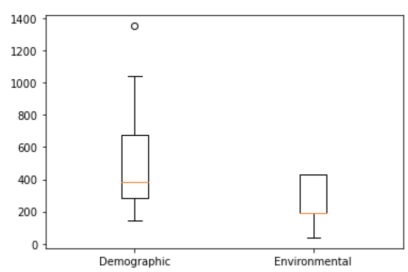
	0	1	2	3	4	5	6	7	8	9	...
0	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	...
1	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	...
2	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	...
3	125.971200	125.971200	125.971200	125.971200	125.971200	125.971200	125.971200	125.971200	125.971200	125.971200	...
4	136.048896	136.048896	136.048896	136.048896	136.048896	136.048896	136.048896	136.048896	136.048896	136.048896	...
5	146.932808	146.932808	146.932808	146.932808	146.932808	146.932808	146.932808	65.303470	146.932808	146.932808	...
6	158.687432	158.687432	158.687432	158.687432	158.687432	158.687432	158.687432	70.527748	158.687432	158.687432	...
7	171.382427	171.382427	171.382427	76.169968	171.382427	171.382427	171.382427	76.169968	171.382427	171.382427	...
8	185.093021	185.093021	185.093021	82.263565	185.093021	185.093021	185.093021	82.263565	185.093021	185.093021	...
9	199.900463	199.900463	199.900463	88.844650	199.900463	199.900463	199.900463	88.844650	199.900463	199.900463	...
10	95.952222	215.892500	215.892500	95.952222	215.892500	215.892500	215.892500	95.952222	95.952222	215.892500	...
11	103.628400	233.163900	233.163900	103.628400	233.163900	233.163900	233.163900	103.628400	103.628400	233.163900	...
12	111.918672	251.817012	251.817012	111.918672	251.817012	251.817012	251.817012	49.741632	111.918672	251.817012	...
13	120.872166	271.962373	271.962373	120.872166	271.962373	271.962373	271.962373	53.720962	120.872166	271.962373	...
14	130.541939	293.719362	293.719362	130.541939	293.719362	293.719362	293.719362	25.786062	130.541939	293.719362	...
15	140.985294	317.216911	140.985294	140.985294	317.216911	317.216911	317.216911	27.848947	140.985294	317.216911	...
16	67.672941	342.594264	152.264117	152.264117	342.594264	342.594264	342.594264	30.076863	67.672941	342.594264	...
17	73.086776	370.001805	164.445247	164.445247	370.001805	370.001805	370.001805	32.483012	73.086776	370.001805	...
18	78.933719	399.601950	177.600867	177.600867	399.601950	399.601950	399.601950	35.081653	78.933719	399.601950	...
19	85.248416	431.570106	191.808936	191.808936	431.570106	431.570106	431.570106	37.888185	85.248416	431.570106	...

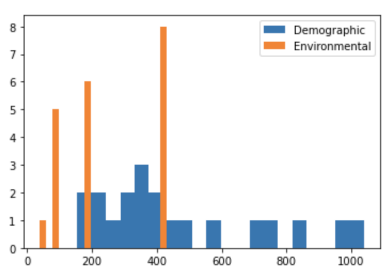
20 rows × 50 columns

[133]:

8	9	...	40	41	42	43	44	45	46	47	48	49
100.000000	100.000000	...	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
108.000000	108.000000	...	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000
116.640000	116.640000	...	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	116.640000	51.840000	116.640000	116.640000
125.971200	125.971200	...	125.971200	125.971200	125.971200	55.987200	125.971200	125.971200	125.971200	55.987200	125.971200	125.971200
136.048896	136.048896	...	60.466176	136.048896	136.048896	60.466176	136.048896	136.048896	136.048896	60.466176	136.048896	136.048896
146.932808	146.932808	...	65.303470	146.932808	146.932808	65.303470	146.932808	146.932808	146.932808	65.303470	65.303470	146.932808
158.687432	158.687432	...	70.527748	158.687432	158.687432	70.527748	158.687432	70.527748	158.687432	70.527748	70.527748	158.687432
171.382427	171.382427	...	76.169968	171.382427	171.382427	76.169968	171.382427	76.169968	171.382427	76.169968	76.169968	171.382427
185.093021	185.093021	...	82.263565	185.093021	185.093021	82.263565	185.093021	82.263565	185.093021	82.263565	82.263565	185.093021
199.900463	199.900463	...	88.844650	199.900463	199.900463	88.844650	199.900463	88.844650	199.900463	88.844650	88.844650	199.900463
95.952222	215.892500	...	95.952222	215.892500	215.892500	95.952222	215.892500	42.645432	215.892500	95.952222	95.952222	215.892500
103.628400	233.163900	...	103.628400	233.163900	233.163900	103.628400	233.163900	46.057067	233.163900	103.628400	103.628400	233.163900
111.918672	251.817012	...	111.918672	251.817012	251.817012	111.918672	251.817012	49.741632	251.817012	111.918672	111.918672	251.817012
120.872166	271.962373	...	120.872166	271.962373	271.962373	120.872166	271.962373	53.720962	271.962373	120.872166	120.872166	271.962373
130.541939	293.719362	...	130.541939	293.719362	293.719362	130.541939	293.719362	58.018639	293.719362	130.541939	130.541939	293.719362
140.985294	317.216911	...	140.985294	317.216911	317.216911	140.985294	317.216911	62.660131	317.216911	140.985294	140.985294	317.216911
67.672941	342.594264	...	152.264117	342.594264	342.594264	152.264117	342.594264	67.672941	342.594264	152.264117	152.264117	342.594264
73.086776	370.001805	...	164.445247	370.001805	370.001805	164.445247	370.001805	73.086776	370.001805	164.445247	164.445247	370.001805
78.933719	399.601950	...	177.600867	399.601950	399.601950	177.600867	177.600867	78.933719	177.600867	177.600867	177.600867	177.600867
85.248416	431.570106	...	191.808936	431.570106	431.570106	191.808936	191.808936	85.248416	191.808936	191.808936	191.808936	191.808936

9.4





```python
print("mean:", model.T[19].mean())
print("median:", model.T[19].median())
print("standard deviation:", model.T[19].std())
print("25 percentile:", model.T[19].quantile(.25))
print("75 percentile:", model.T[19].quantile(.75))
print("min:", model.T[19].min())
print("max:", model.T[19].max())
print("range:", model.T[19].max() - model.T[19].min())
print("interquartile:", model.T[19].quantile(.75) - model.T[19].quantile(.25))
```

```
mean: [486.00445113]
median: 381.7668622295612
standard deviation: 289.28875147986827
25 percentile: [285.64680669]
75 percentile: [674.8470935]
min: [144.03944629]
max: [1353.66070531]
range: [1209.62125902]
interquartile: [389.20028682]
```

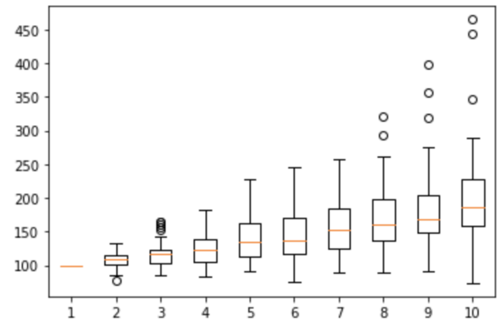
```python
print("mean:", model2.T[19].mean())
print("median:", model2.T[19].median())
print("standard deviation:", model2.T[19].std())
print("25 percentile:", model2.T[19].quantile(.25))
print("75 percentile:", model2.T[19].quantile(.75))
print("min:", model2.T[19].min())
print("max:", model2.T[19].max())
print("range:", model2.T[19].max() - model2.T[19].min())
print("interquartile:", model2.T[19].quantile(.75) - model2.T[19].quantile(.25))
```

```
mean: 289.00923248673
median: 311.689520936425
standard deviation: 149.66297469874942
25 percentile: 191.80893596087688
75 percentile: 431.5701059119731
min: 16.83919328051594
max: 431.5701059119731
range: 414.73091263145716
interquartile: 239.7611699510962
```

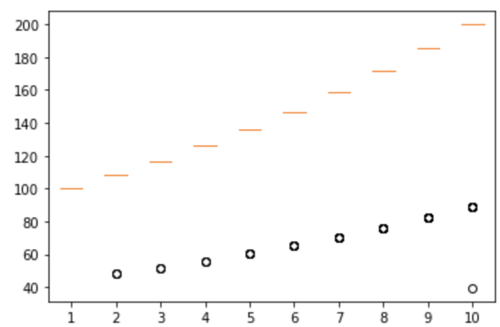
The demographic stochastic model had a much more spread as well as higher mean and median than the environmental stochastic model. This is most likely due to the fact that after 20 years, most simulations had encountered a disaster, causing the population rate to decline heavily.

9.5

```
[347]: plt.boxplot([model.T[i] for i in range(0, 10)])
       plt.show()
```




```
[349]: plt.boxplot([model2.T[i] for i in range(0, 10)])
       plt.show()
```



There is very little variation from the mean at all with only a few outliers each time.

9.6

The environmental impact is much more impact on the population over time. As time goes on, a population will most likely experience a catastrophe that will decrease the population significantly.

10

I am going to work ont eh final project with Evan Coons, Jessie Baker, and Ronan Nayak.