PredatorPreyPyCX

April 4, 2025

1 PyCX Predator Prey Model

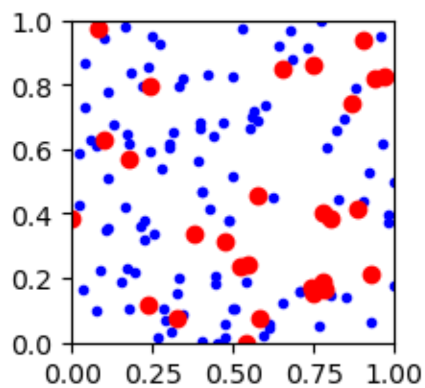
Adapted from Hiroki Sayama's PyCX module. For more information about the model, please see Sections 4.6 and 19.4 of Sayama's book (Introduction to the Modeling and Analysis of Complex Systems).

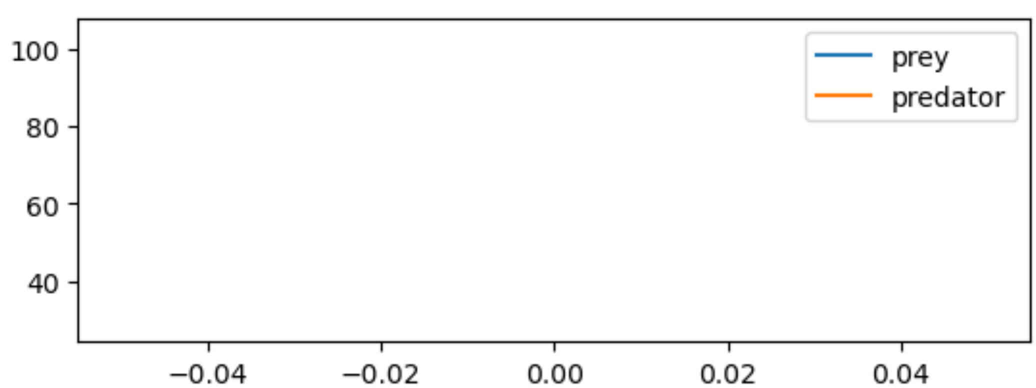
1.0.1 Load needed packages

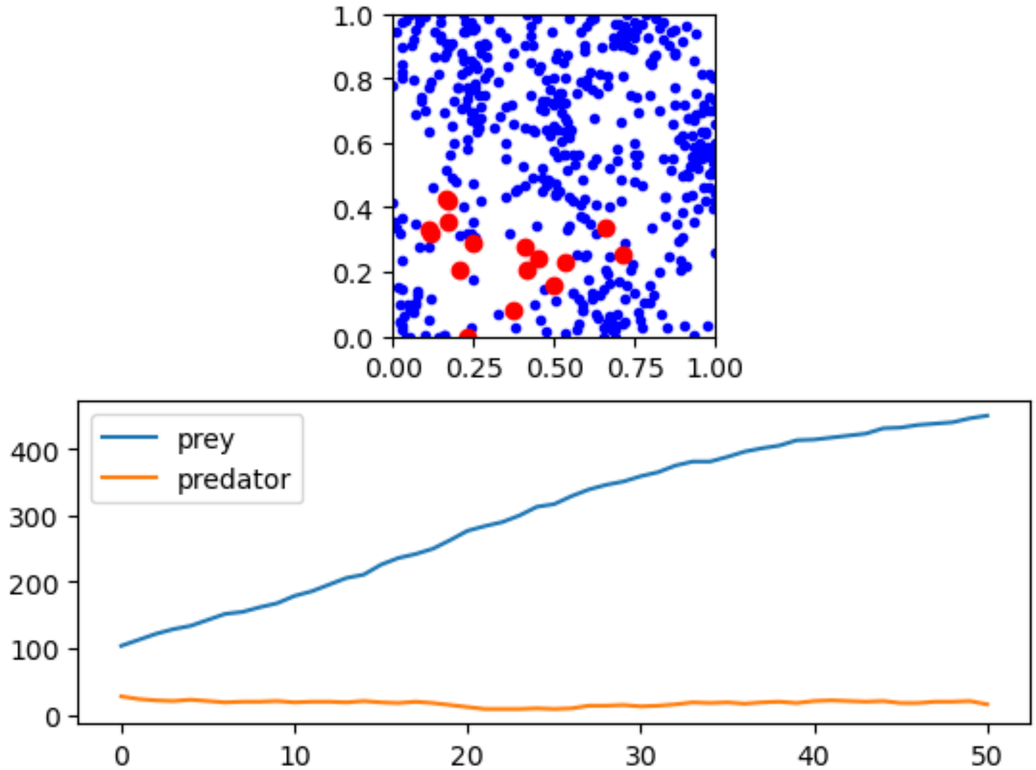
Below includes an example of how to install and load `pyDOE` for Latin hypercube sampling.

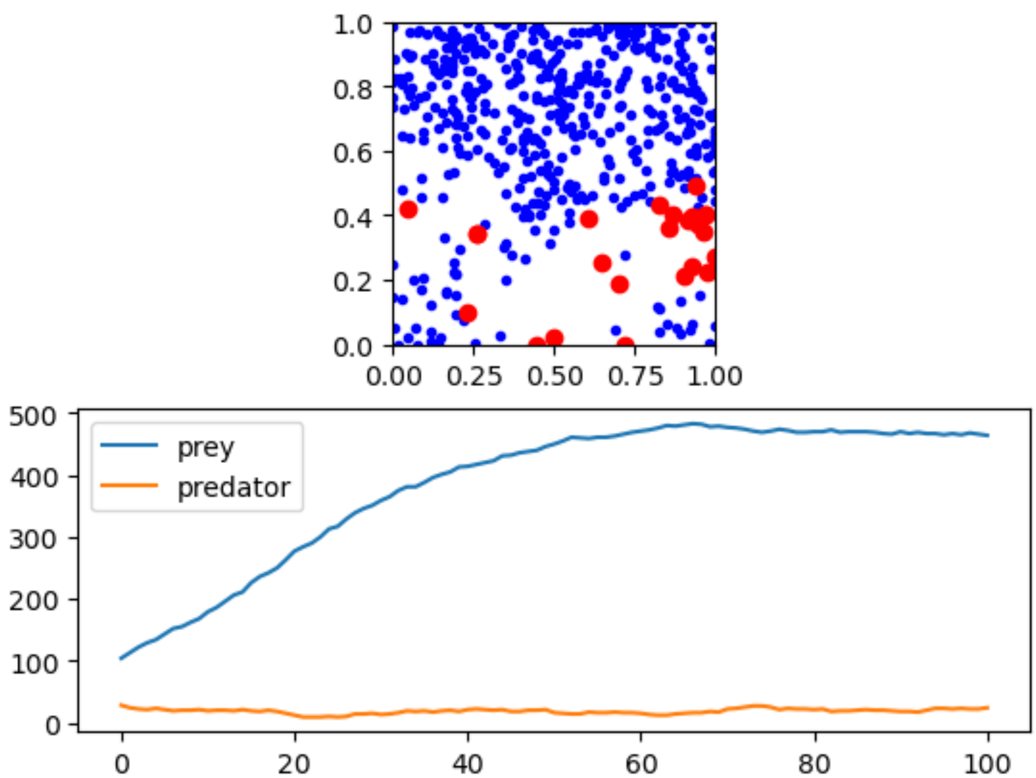
1.0.2 Set up model parameters

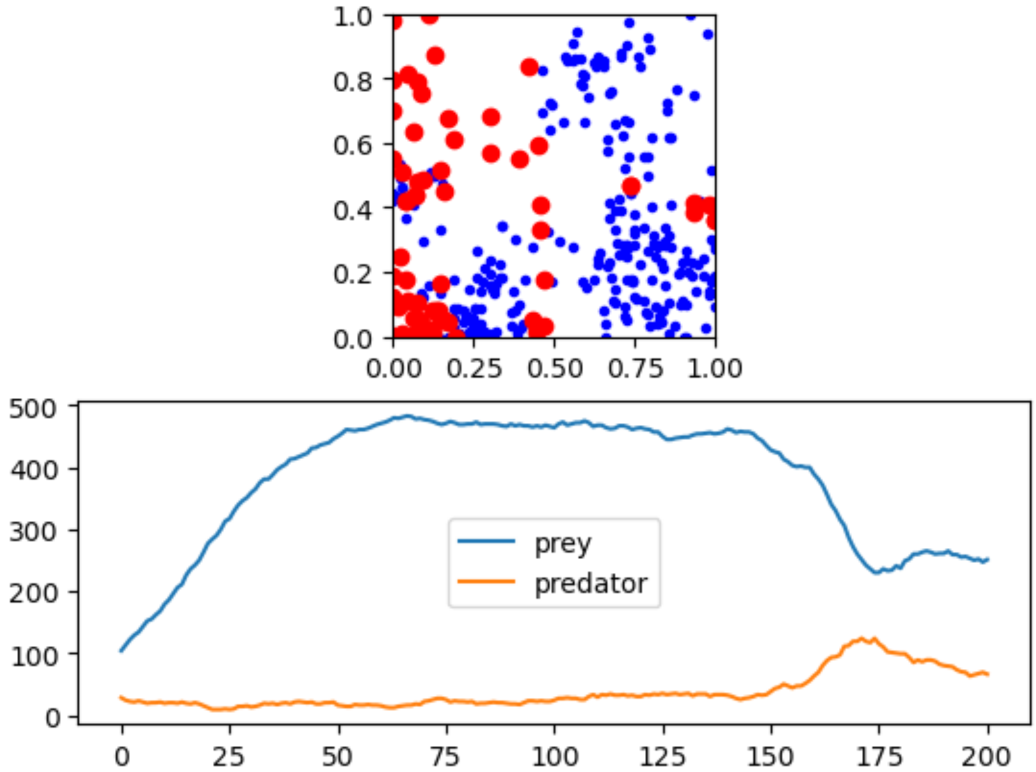
2 1A - Exploring the Model

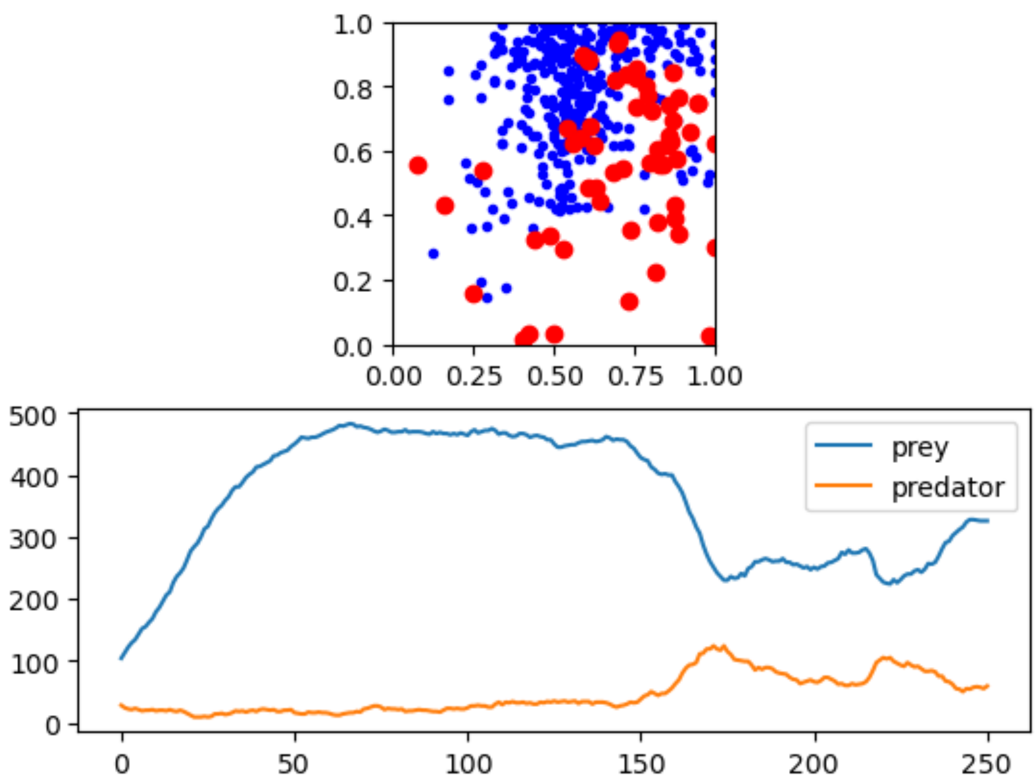


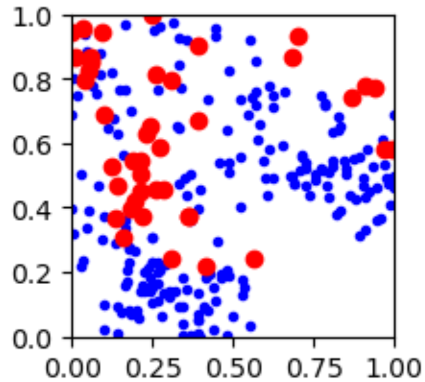


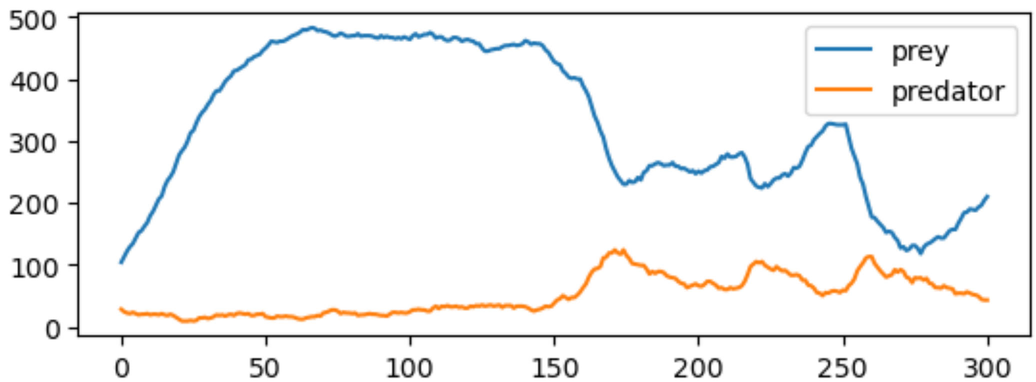

1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
prey
predator
400
300
200
100
0
0
10
20
30
40
50

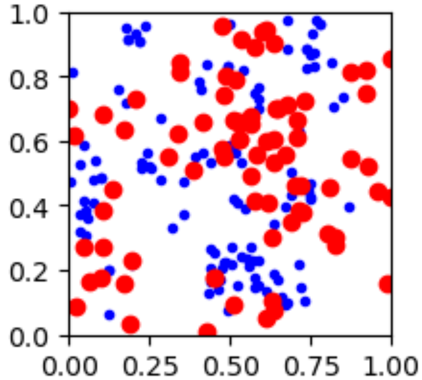

1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
prey
predator
500
400
300
200
100
0
0
20
40
60
80
100

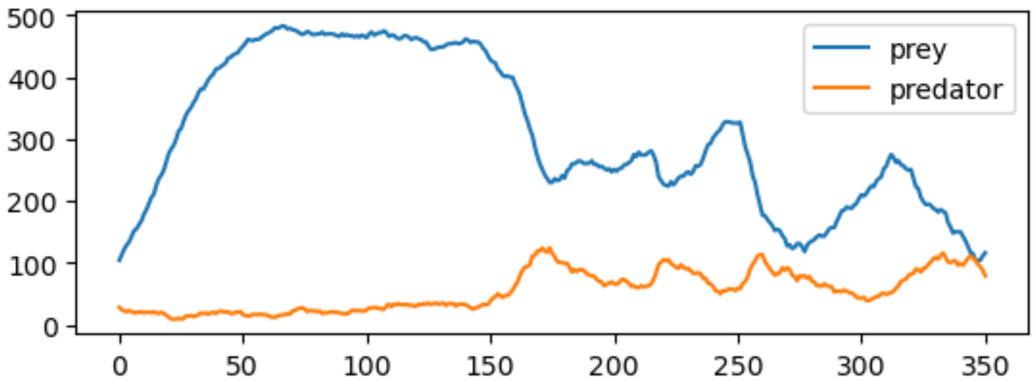

prey
predator


prey
predator


1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00


prey
predator
500
400
300
200
100
0
0
50
100
150
200
250
300

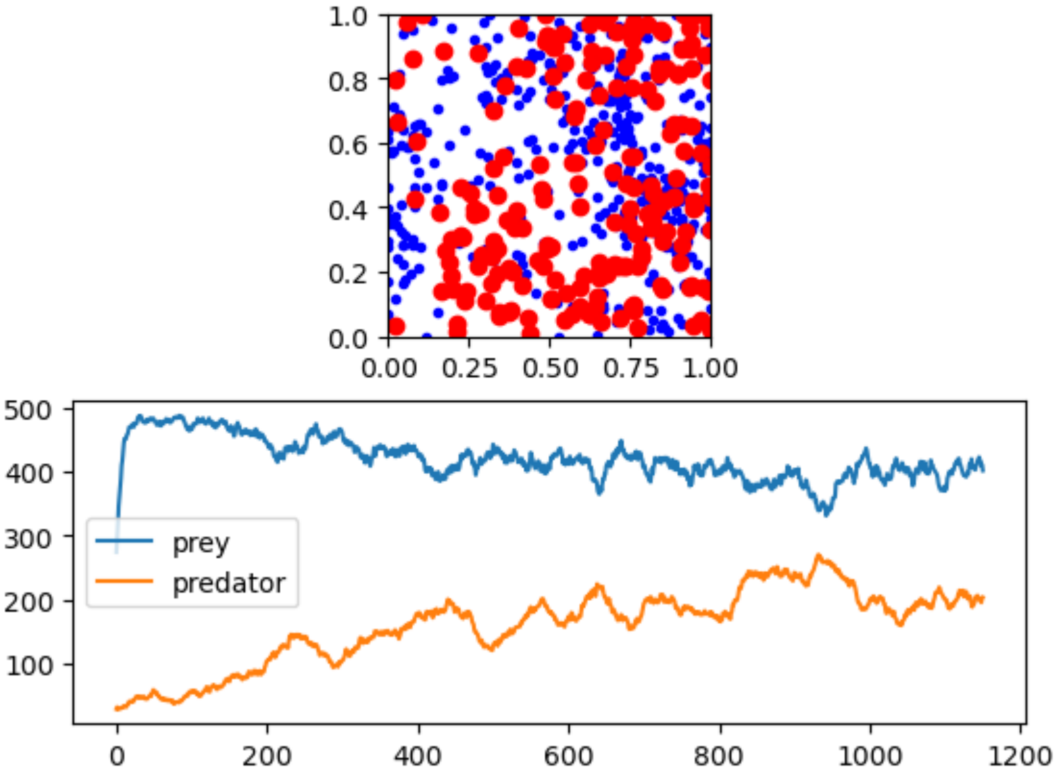


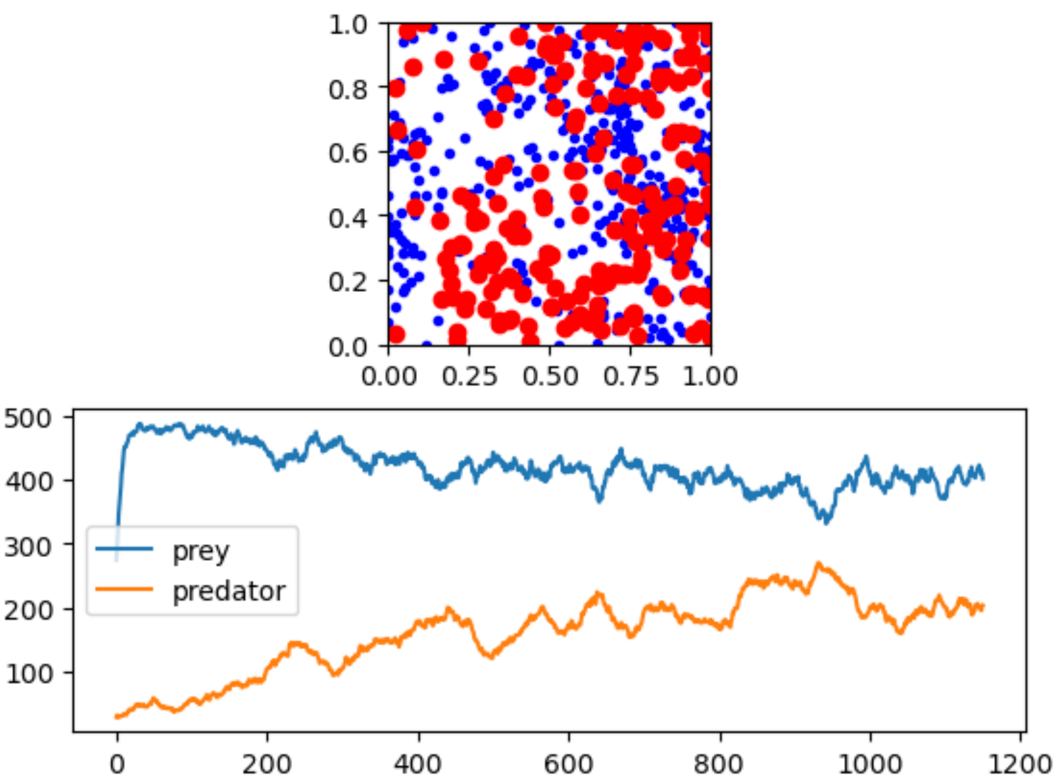


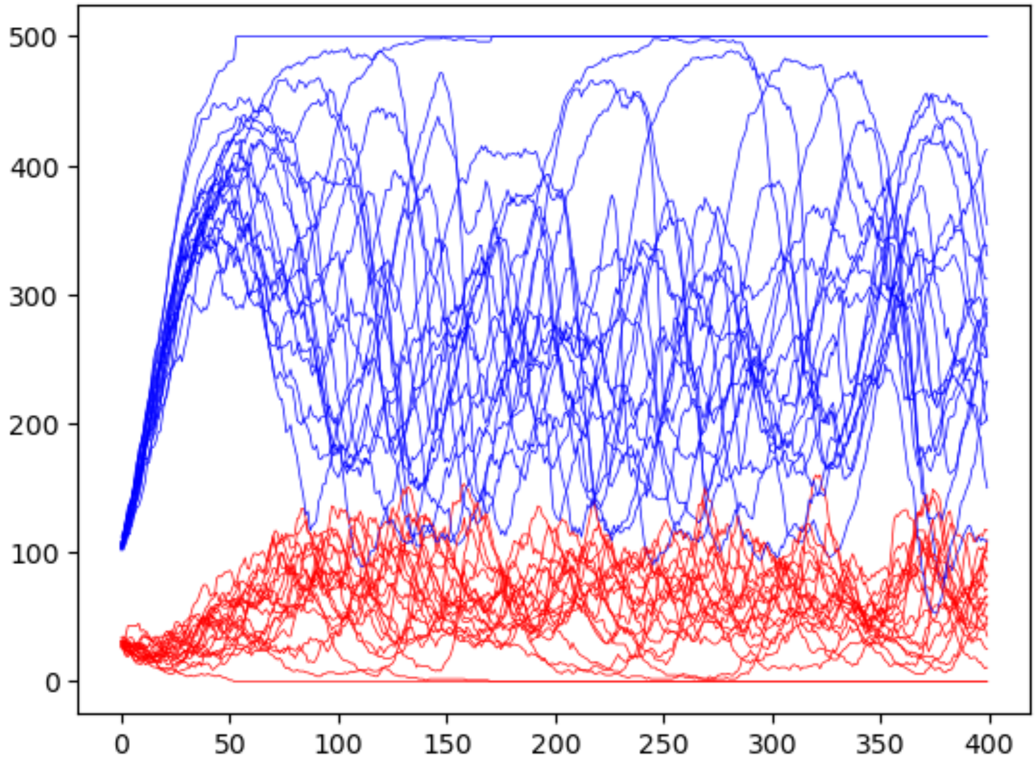
2.1 Discussion

The number of rabbits seems to oscillate significantly , while the number of foxes is relatively stable. (when more rabbits than foxes)

When numbers are closer but same order, there is more pronounced oscillation, but the behavior remains similar. The predator and pray lines very rarely cross. Additionally, it looks like the dynamics is for the rabbit population to thin out considerably in areas that are dense in foxes, eventually accumulating around the grid corners, and eventually growing back towards the center once the fox population has thinned out appropriately. In our models convergence is unlikely unless the predator population dies out, while in Sayama 4.6, both populations tend to stabilize (to non-zero values) over time.


prey
predator


prey
predator



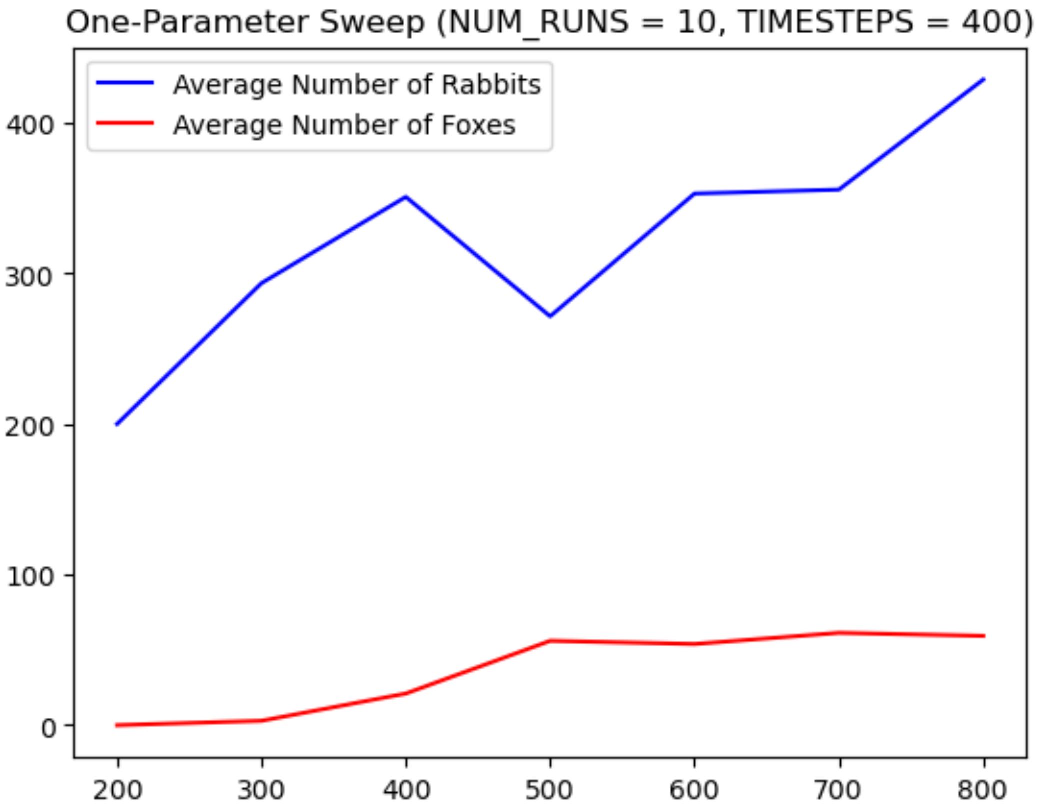
3.1 Discussion

Plotting out different trajectories suggest that there are mainly two results from the simulation: - Rabbits -> NR, Foxes-> 0 - Rabbit & Fox population just oscillates

Additionally, it's worth noticing that the rabbit population is almost always above the fox population.

4 Problem 1C - One-Parameter Sweep

```
Running for NR = 200…
        Running simulation #0…
        Running simulation #1…
        Running simulation #2…
        Running simulation #3…
        Running simulation #4…
        Running simulation #5…
        Running simulation #6…
        Running simulation #7…
        Running simulation #8…
        Running simulation #9…
Done
```

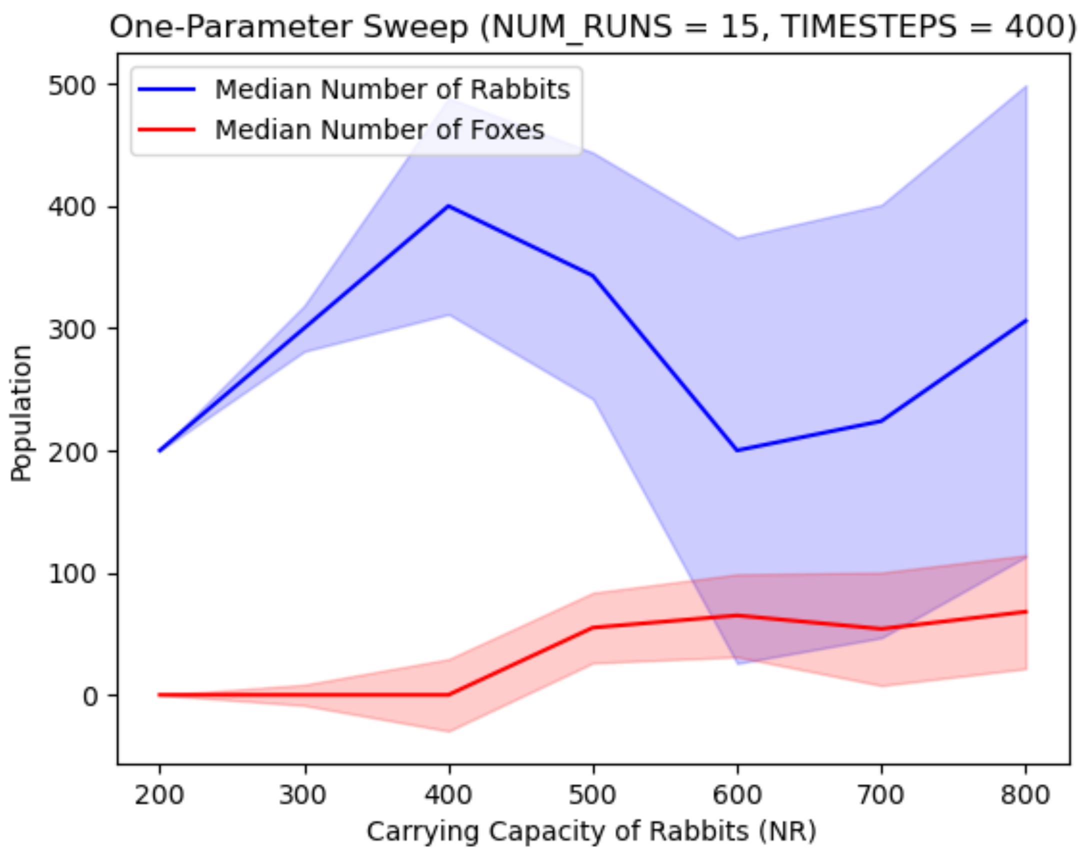


4.1 Discussion

The trend seems to be for both population to increase as the initial rabbit count increases. This makes sense: notice that the minimum oscillation point for the rabbit population also tends to be the maximum oscillation point for the fox population. Starting with more rabbits, the fox population will increse much faster, setting this point to be higher. The average is approximately the midline of both those oscillations, so higher capacity should drive the average of both populations up. This definitely could be mitigated by variation in other parameters, such DR and DF.

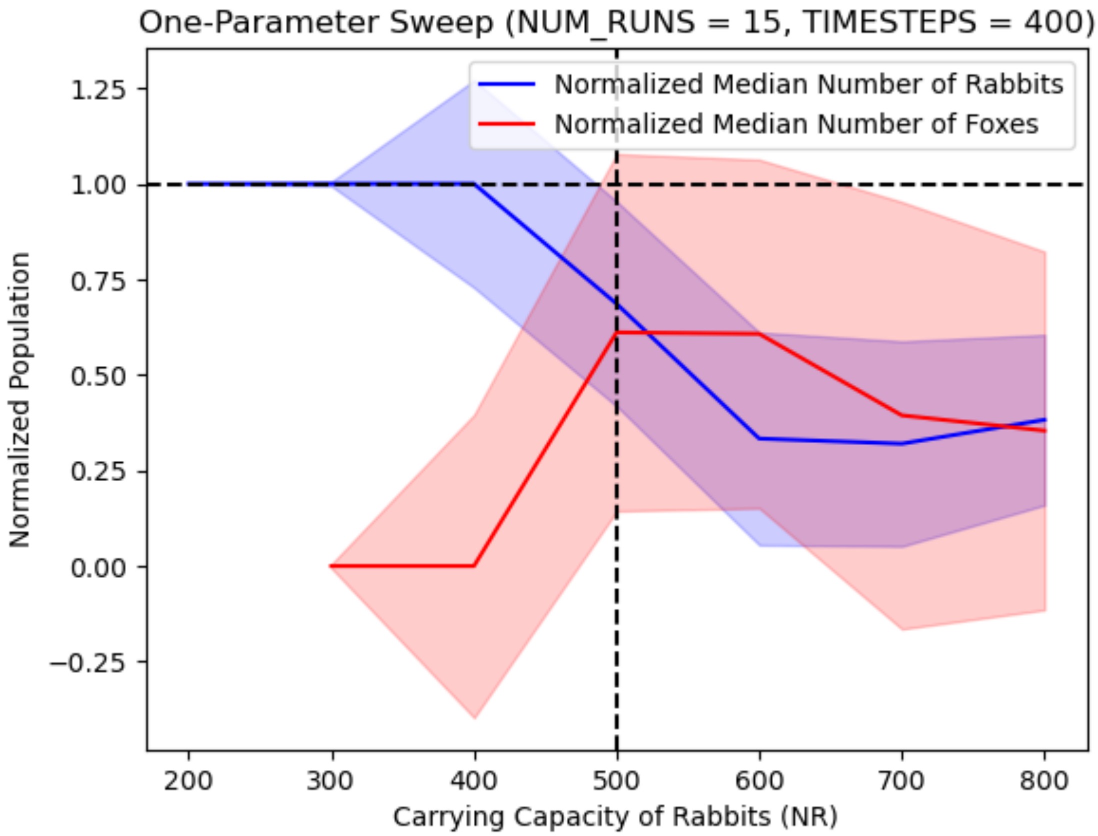
On the other hand, there are many interesting characteristics that the mean final population fails to account for, such as the size of the oscillations in population across runs. To have an idea of this, measure, the standard deviation could be interesting. Another model output worth considering is the *median* final population. That could be especially interesting – the issue with the mean is that if a few runs end at a value which is much higher than a standard average (for example in case of predator extinction), it will pull the curve up. We decide to graph both the final median (solid line) and standard deviation across runs (dashed line).

```
Running for NR = 200…
        Running simulation #1…
        Running simulation #2…
        Running simulation #3…
        Running simulation #4…
```



4.2 Discussion

This is a pretty interesting graph! We can see that for low carrying capacity the outcome was the same across *all* runs – predator extinction. We can also see that the standard deviation of the rabbit population is much higher than that of the fox population. As a test, let's look at the normalized version of this graph for the prey:



4.3 Discussion

From the normalized viewpoint, not only are population sizes *comparable* over time, but it looks like the fox population has a higher standard deviation. We see the inverse correlation that we would expect between the final number of prey and predator play out.

Lab 4

April 4, 2025

1 1 - Run the Sweep.

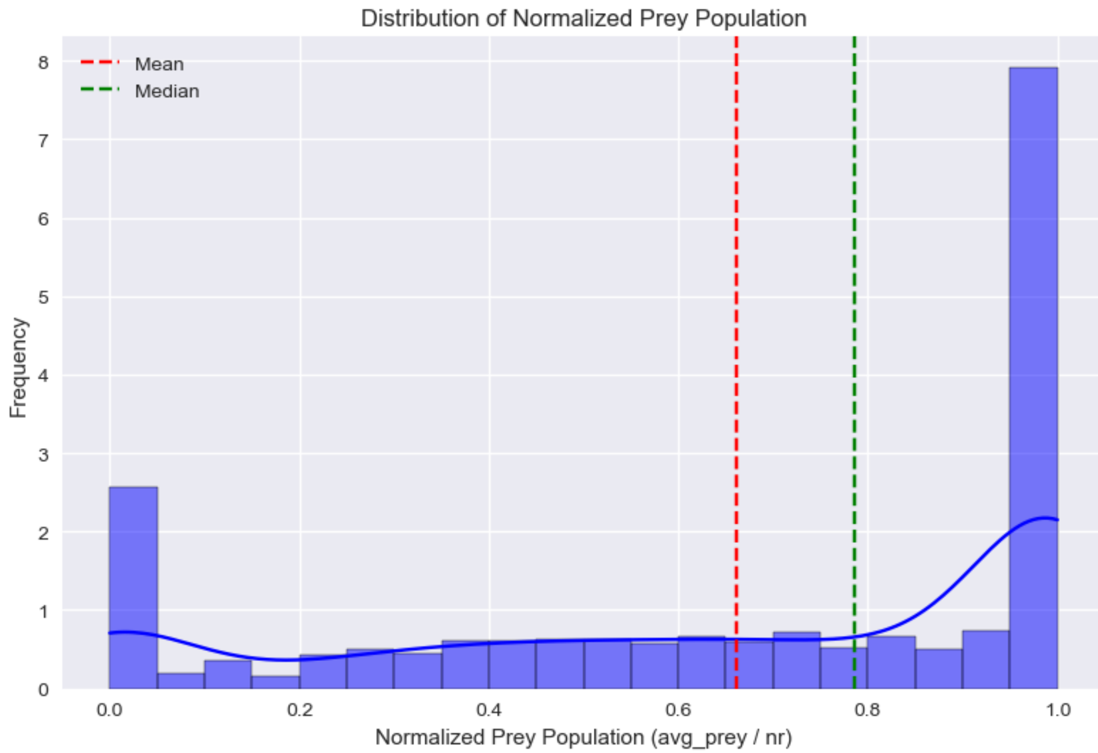
The data is saved inside a Pandas DataFrame

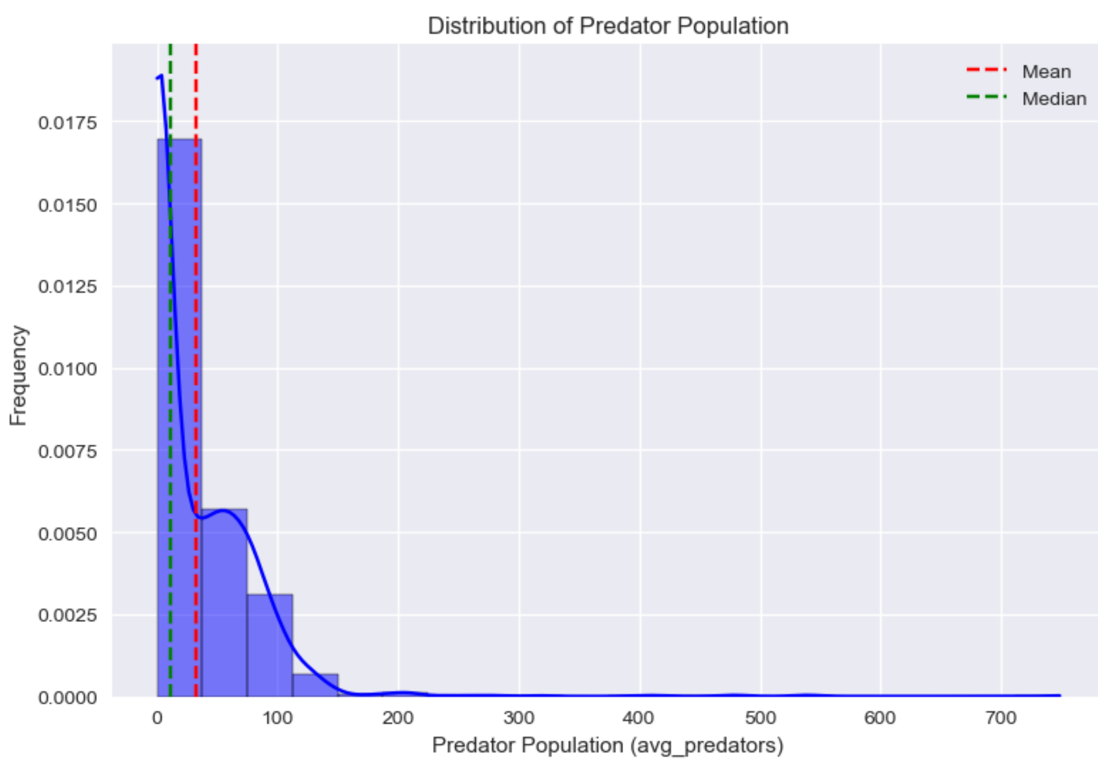
2 2 - Plot results

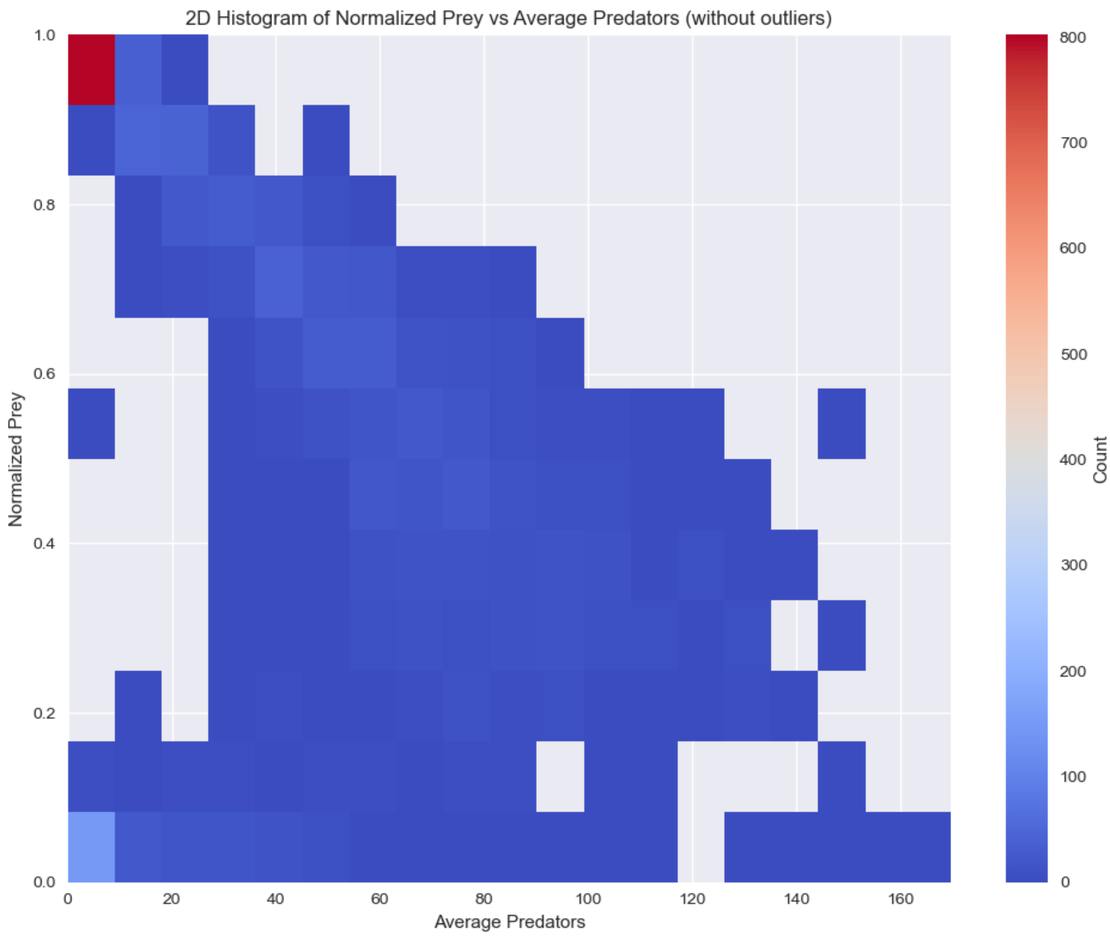
	sample	nr	dr	df	rf	avg_prey	std_prey	avg_predators	\
0	0	432	0.85250	0.146250	0.35225	432.0	0.0000	0.0	
1	1	305	0.84175	0.170625	0.50550	305.0	0.0000	0.0	
2	2	655	0.60925	0.138000	0.66700	364.7	12.0208	83.2	
3	3	598	0.59050	0.042500	0.54525	83.9	86.1256	22.7	
4	4	314	0.92025	0.028375	0.29525	111.0	46.1034	55.8	

	std_predators	normalized_prey
0	0.000000	1.000000
1	0.000000	1.000000
2	0.565685	0.556794
3	8.626700	0.140301
4	7.353910	0.353503

2.1 Histograms



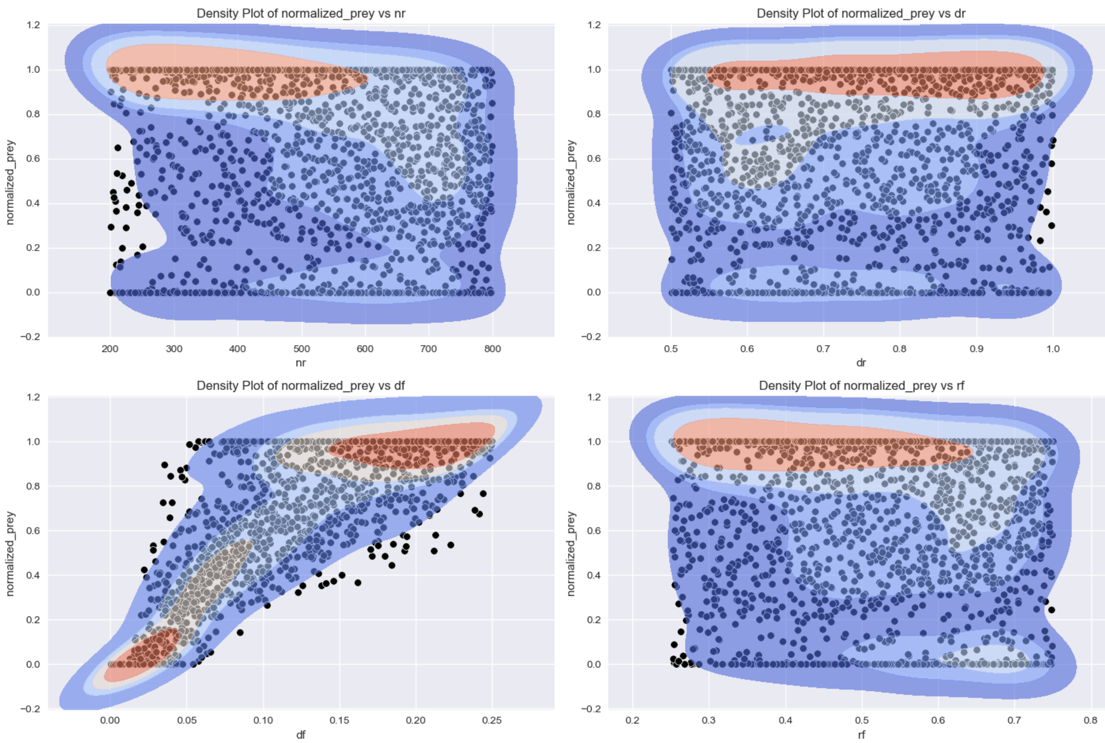




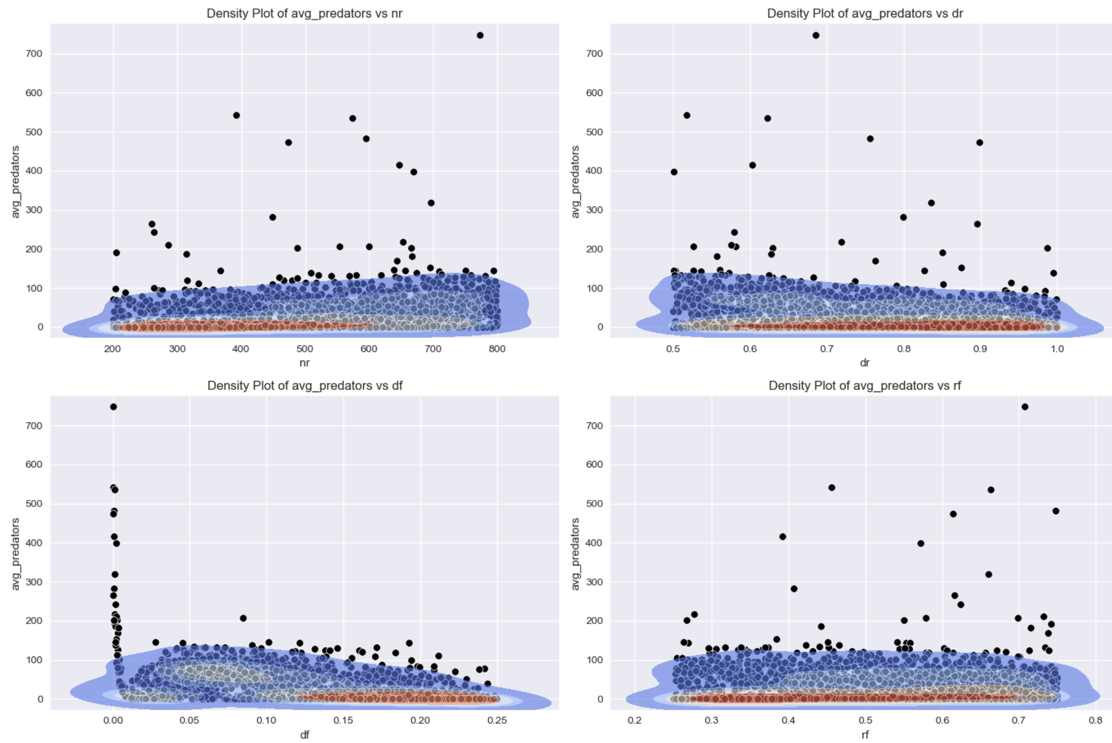
2.2 Heatmaps

We plot heatmaps of each parameter against population averages.

2.2.1 Normalized Prey Population vs NR, DF, DR, RF

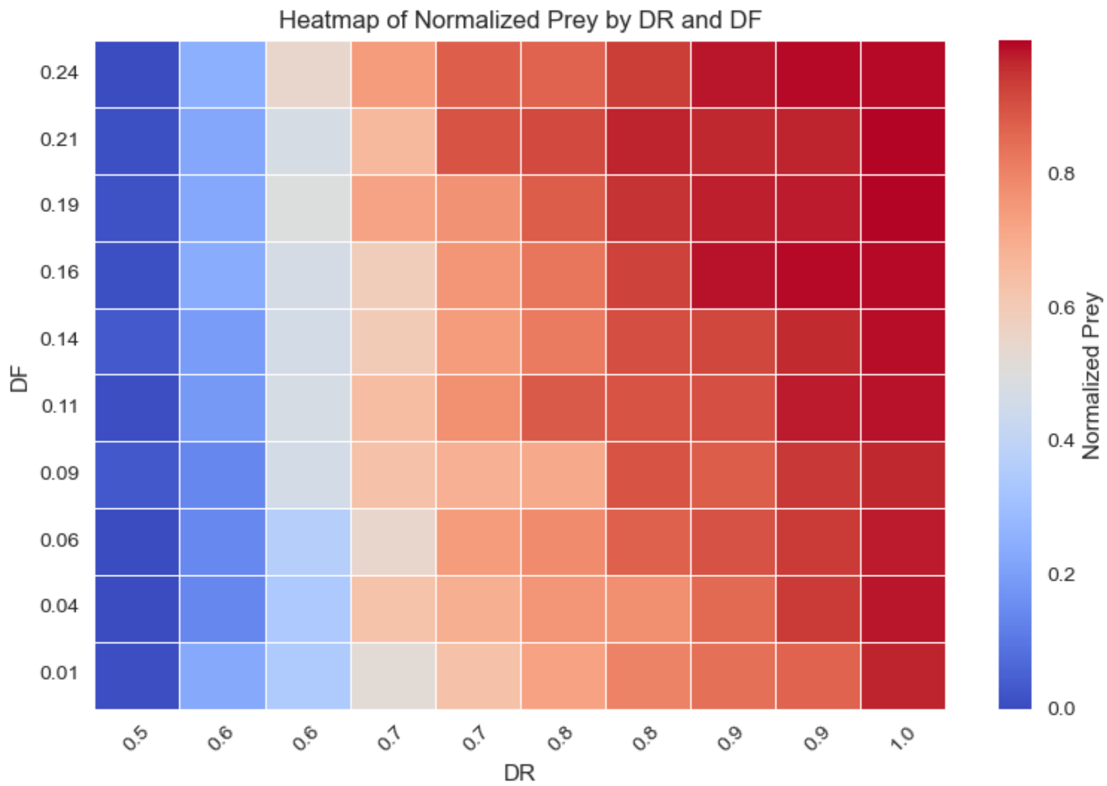


2.2.2 Predator Population vs NR, DF, DR, RF

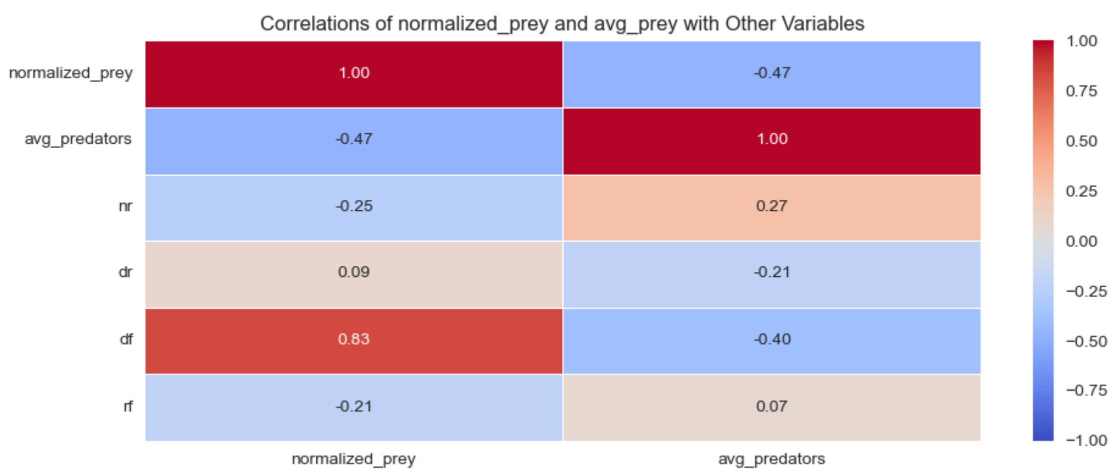


2.3 Two Parameter Heatmaps

```
/var/folders/mx/slrd_4mx3r71wjgg45nvz53w0000gr/T/ipykernel_55633/35344035.py:10:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  heatmap_data = df.groupby(['dr_bin',
'df_bin'])['normalized_prey'].mean().unstack()
```



2.4 Correlation Coefficients Heatmap



3 Discussion

3.1 1D - Sampling parameter space and uncertainty quantification

The data behaves more or less as expected. One very surprising observation is that the `normalized prey` vs `df` scatterplot has a relatively high density near the bottom left corner! We see a very strong correlation between `df` and `avg_predators` in the correlation heatmap, so this bottom-left accumulation has to come from a mitigating parameter – or combination thereof. From the 2D parameter heatmap, it seems like this pool may be caused by high values of NR.

Just as before, looking at means can distort data in ways that make it hard to spot trends. In particular, it is pretty clear from scatter plots that fox populations seem concentrated towards the bottom of the population interval across all parameters. The median might be a more faithful output variable to look at. The way the code is currently setup, the sweep doesn't return historical data, so we would have to modify our C++ code and re-run the simulation.