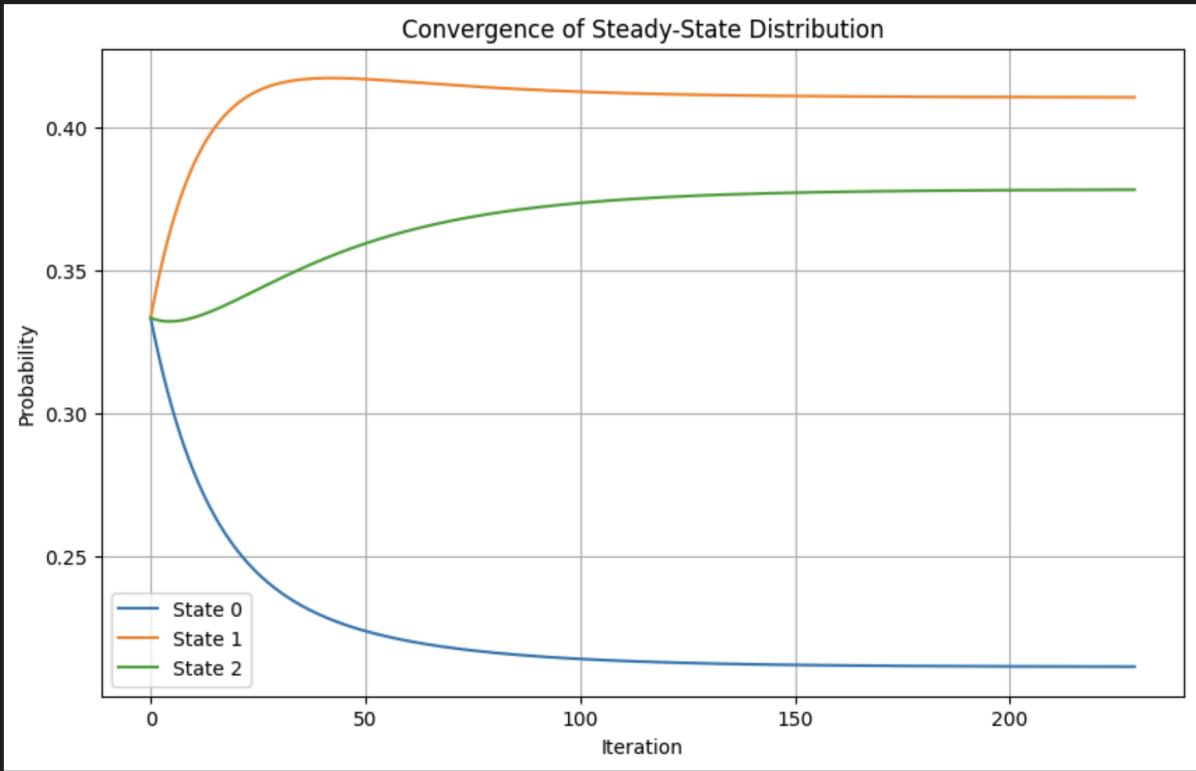
1) What is the steady state distribution of your Markov Chain(s) and over what period of time is it achieved? How does your steady state convergence change as a function the number of states?

For the normal 3 step:

```
Converged at step 229
```

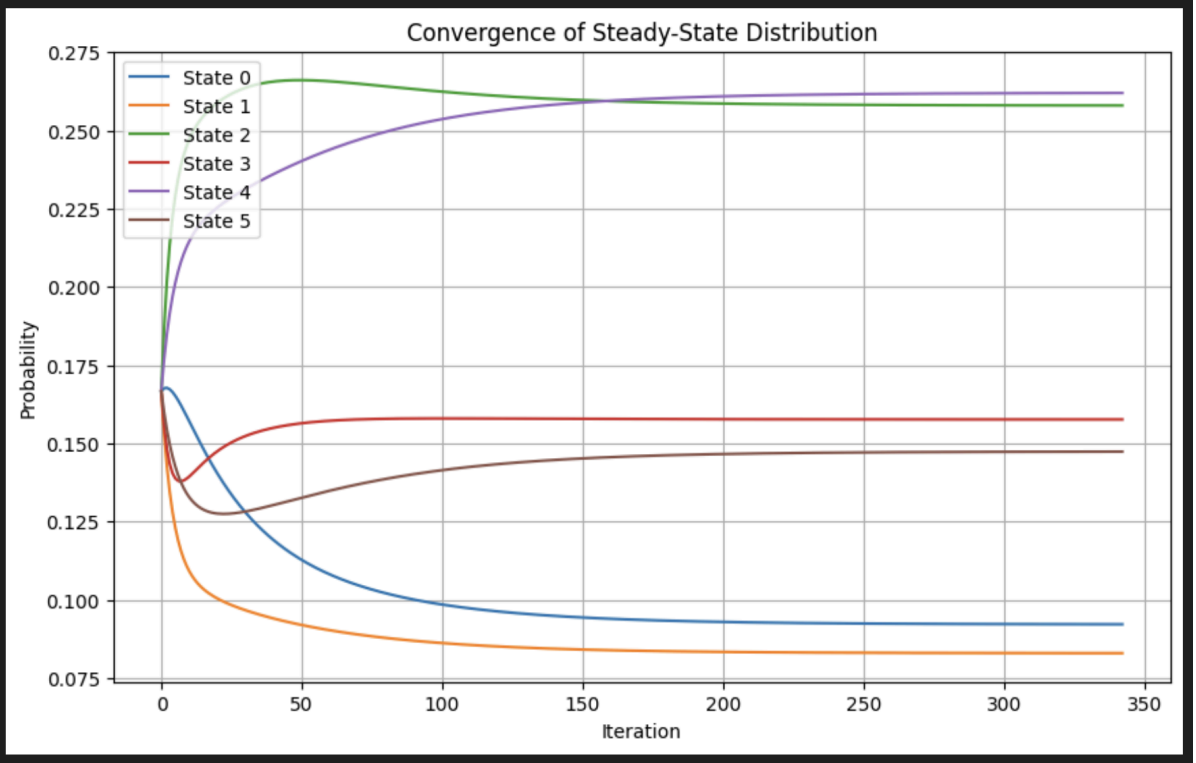



```
steady state distribution

[0.21129532 0.41051966 0.37818502]
```

For the experiment with 6 different states:

```
Converged at step 342
```

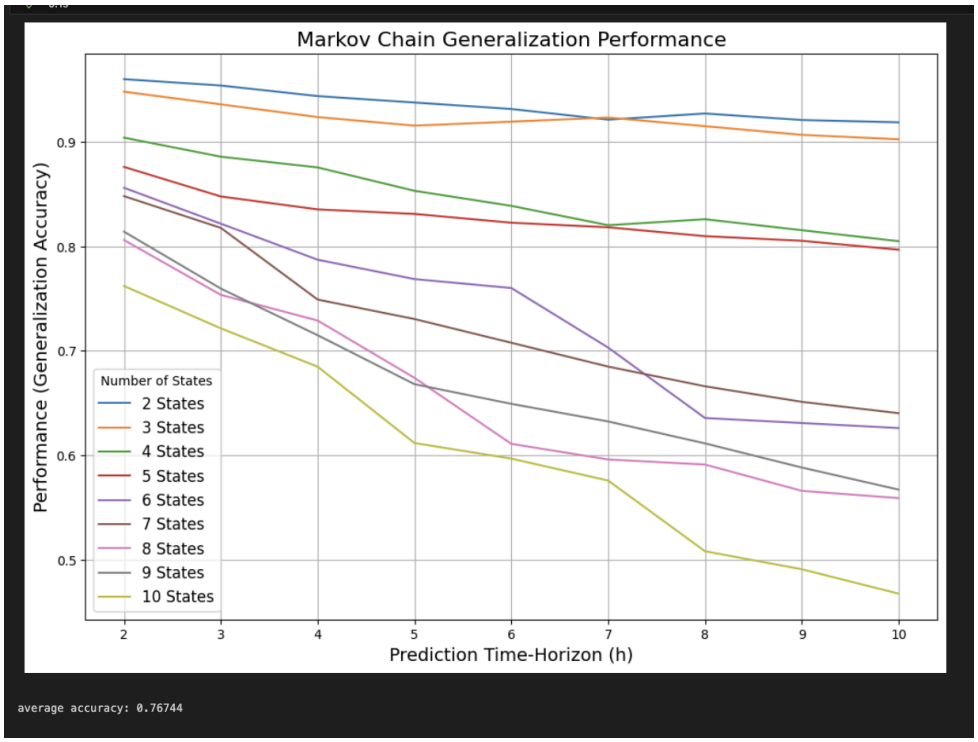


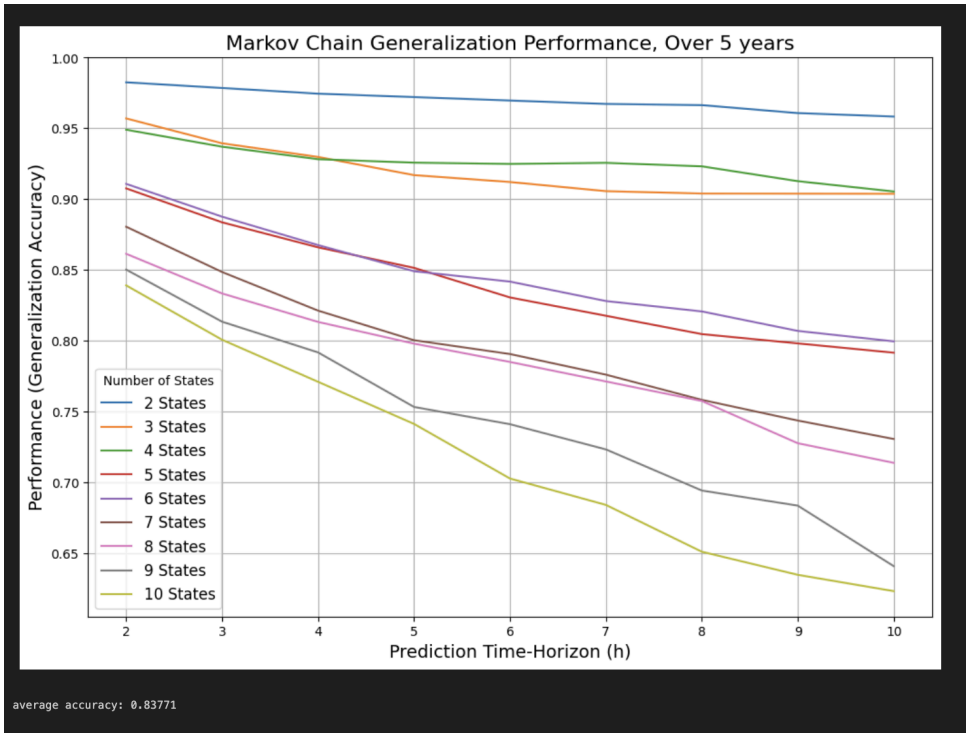

```
steady state distribution

[0.09216295 0.08293065 0.25796817 0.15762937 0.26196393 0.14734493]
```

Analysis: We can see the it takes more than a hundred more iterations to reach convergence when I increased the number of states from 3 to 6. This is due to the fact that as the number of states increases, the system has more complexity and a larger state space to explore. This naturally requires more transitions for the Markov Chain to stabilize.

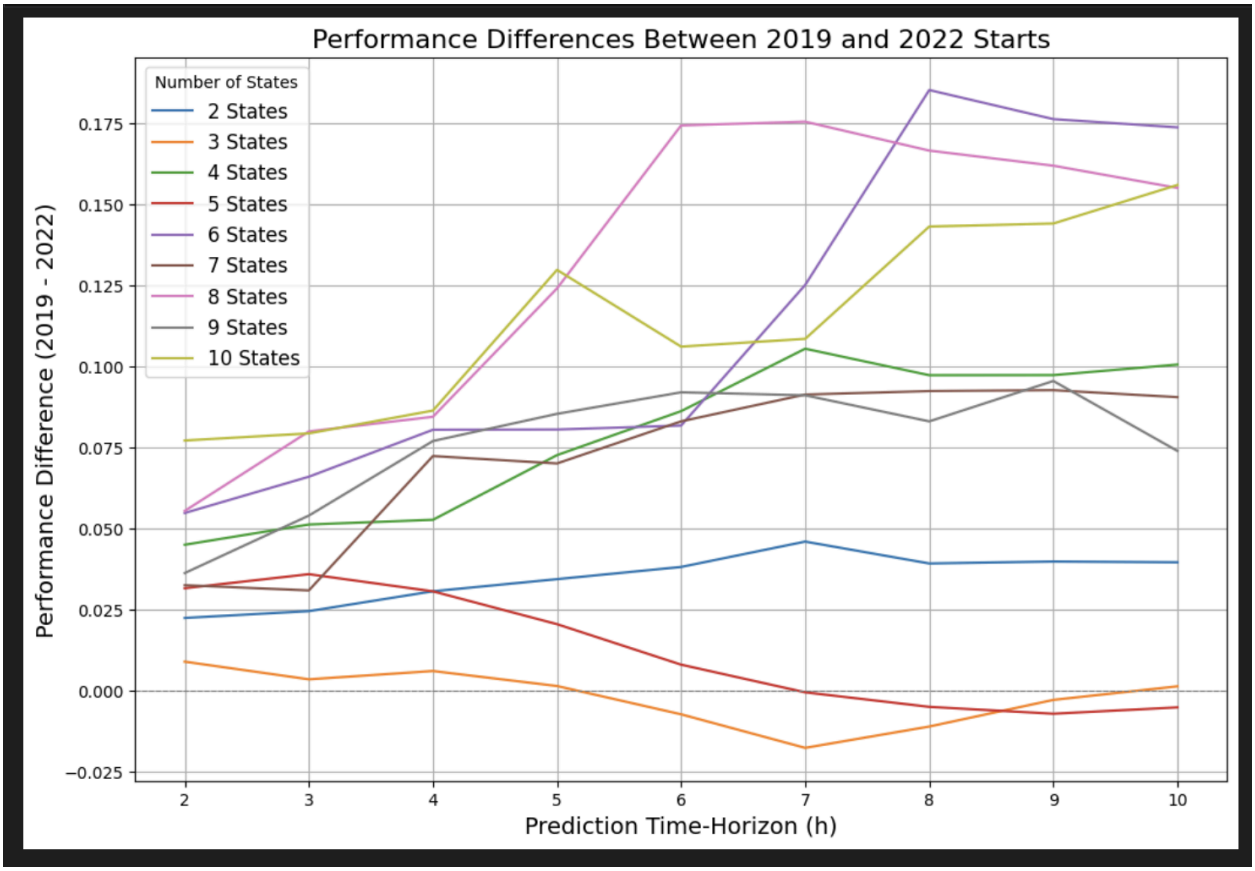
2) How does the Markov Chain's generalization change as a function of the number of states (1,...,10) and the prediction time-horizon (time steps h=1,...,10). You can report this as a graph (multiple line chart) or a table.





For both the five year, and the 2 year case we can see a loose but consistent trend that performance worsens with the addition of both longer time horizons, and states. This makes sense as it's harder to generalize for both. This is why it's important to find a good tradeoff between longer time horizons allowing for more temporal variation while also ensuring that accuracy of the system is also considered. Another tradeoff is how more states can catch a lot of the noise within the system.

3) How does increasing the reporting period to 5-years impact your Markov Chain's prediction performance?




```
comparison_results.describe().loc[["mean","std"]]
```

✓ 0.0s

	num_bins	horizon	performance_2019	performance_2022	performance_diff
mean	6.000000	6.000000	0.837714	0.767444	0.070270
std	2.598076	2.598076	0.092894	0.130468	0.052337

The increase of the reporting period from 2022 to 2019 led to an increase in general performance by on average 6%. In addition the variance decreased by about 4%. This indicates that the longer reporting period did have a significant effect on improving the performance of the system. Most notably performance improved significantly for the markov calculations with more states, and or time periods. While the performance for the already “high” accuracy states and prediction time horizons does not increase significantly. This shows that the addition of data is most helpful in allowing us to use more states, or longer horizon’s while not significantly improving the already high accuracy results.

Reflection

a) How did you go about doing the work?

I first looked over my class notes that helped me remind myself about what a steady state is.. Once I felt like I had a solid understanding, I came to the realization that the best strategy would be to use the allclose numpy function (as I've used it before for a basketball ml project), I chose a standard tolerance and had my steady state function. After that I closely read the instructions provided to make the markov chain prediction function. To solve the problem I split it into three sub-tasks, a) State Discretization, b) Matrix Initialization, c) State Landing, and d) Evaluation. Since many of these had already been shown in the example code I didn’t have to do a lot of trial and error. Finally I created the run experiment where I just iterated over the different state, and time horizon ranges, while applying the previously created function within the time block. For plotting and better analysis I also created two other functions that helped me visualize the difference between states and datasets.

b) Were there any surprises?

There weren’t really any major surprises, I can only think of the fact that before the experiments I had the idea that more states would create a more accurate model when in reality it’s the opposite. This helped have a better conceptual understanding of markov chains.

• What did you learn from this experiment?

For this experiment, I learned…

- how to get stock data from the yahoo finance api
- A formal definition of the programing structure of pairwise iteration.
- How to be proficient with using numpy arrays as matrix
- What a transition matrix is
- How to create a h-step matrix
- Most importantly about how to programmatically implement a markov chain in python