**Modeling fishing in “Animal Crossing: New Horizons”**

by India Lindeman

**Introduction**

The peaceful island world of Nintendo’s “Animal Crossing: New Horizons” is the perfect place to pass time during quarantine. Players can catch fish and bugs, design and upgrade their homes, plant trees and flowers, and create the town of their dreams. A viral sensation, the game has sold close to five million copies within the first month of its release (Knight). While the game is praised for its "laid-back," “relaxing” nature (Greenwald), certain elements have the potential to frustrate players. Among these features is the fishing system.

**An Overview of Fishing in “Animal Crossing”**

“New Horizons” features 80 different fish that players can catch throughout the year. Fish will spawn (that’s videogame-speak for “appear”) in the ocean, by the pier, in the river, in ponds, and under waterfalls. Which fish are available at a given time depends on the time of year and the hour of the day. For example, the softshell turtle is only available from 4pm-9am in August and September in the river. Other fish, like the goldfish, are available all year long at any time of day in their respective location (in this case, the pond).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TIME OF DAY** | **LOCATION** | **SEASON** | **PRICE** | **SIZE** |
| 4am-9am  9am-4pm  4pm-9pm  9pm-4am | River  River mouth  Pond  Waterfall  Ocean  Pier | Availability changes monthly | 100-15,000 bells | Extra small  Small  Medium  Large  Extra Large  Giant |

Figure 01: The criteria for each fish in “Animal Crossing: New Horizons.”

**The Experiment**

As I was playing “New Horizons,” I found myself catching an exorbitant amount of sea bass. While larger fish are generally the most profitable, the sea bass proves the exception: at only 400 bells (2,600 less than the next fish in the “large” criteria, according to data compiled by Julia Lee at Polygon), the sea bass is one of the least profitable ocean fish. My frustration with the spawn rate of the sea bass led me to ask, “Just what percentage of fish I catch are actually sea bass?”—so I decided to find out. For ten days, I “caught” fifty fish per day and recorded the results.

**Parameters and Assumptions**

1. **Location**: This data and the accompanying simulation only account for ocean fish. I limited myself to a single location so I would be able to create a more accurate representation of the fish spawn rates for one area rather than a less accurate representation of the spawn rates for every location.
2. **Time of year:** Because the availability of fish changes on a month-by-month basis, this data and the accompanying simulation only apply for the month of May.
3. **Time of day**: Likewise, since the availability of fish changes throughout the day, I kept my hours of data collection between 9am-4pm. For this reason, the barreleye is not included in the data, since is only available outside of the hours I had selected.
4. **Weather**: Unlike the other ocean fish, the coelacanth only spawns when there is rainfall in the game. While I did catch a few coelacanths during the period when I was collecting data, I decided not to include them, since the probability of rain is a separate variable that would require further research.
5. **Independence**: This project assumes that the spawn rates of fish are independent of one another. That is, the probability of the next fish that a player catches is not affected by the previous fish that he or she caught. This also leads to the assumption that if the player “misses” a fish (by scaring it away or simply choosing not to catch it), this choice will have no impact on which fish will spawn next. For this reason, I did not record the number of “misses” I made when collecting data.

**Results**

After ten days of taxing data collection (i.e., sitting in bed and pressing the “A” button on the Switch for hours), I ended up with the following results:

|  |  |  |
| --- | --- | --- |
| **FISH** | **COUNT** | **PERCENT** |
| Anchovy | 16 | 3.20% |
| Barred knifejaw | 24 | 4.80% |
| Boot | 3 | 0.60% |
| Butterfly fish | 26 | 5.20% |
| Can | 5 | 1.00% |
| Clownfish | 19 | 3.80% |
| Great trevally | 1 | 0.20% |
| Horse mackerel | 99 | 19.80% |
| Mahi-mahi | 1 | 0.20% |
| Oarfish | 5 | 1.00% |
| Olive flounder | 37 | 7.40% |
| Red snapper | 27 | 5.40% |
| Rock | 7 | 1.40% |
| Sea bass | 96 | 19.20% |
| Sea horse | 34 | 6.80% |
| Squid | 49 | 9.80% |
| Surgeonfish | 8 | 1.60% |
| Tire | 4 | 0.80% |
| Zebra turkeyfish | 39 | 7.80% |

Figure 02: The number of each fish I caught during my period of data collection.[[1]](#footnote-2)

As I anticipated, the spawn rate of the sea bass was considerably higher than most other fish; the only fish that trumped it was the horse mackerel, a small fish worth even less than the sea bass. All other large fish had spawn rates under 10%.

**The Simulation**

Once I had collected my data, I decided to create a simulation to accompany it. The program I wrote simulates “catching” 500 fish using a weighted random number generator. I could have programmed the simulation in Python or Java or even Matlab, but I decided to create a static webpage with Javascript and HTML/CSS so I could create a friendlier user interface. The program doesn’t require any input from the user besides clicking a button to run the simulation. It can be run an infinite amount of times.

I did not take the time to make the webpage fully responsive; it is not optimized for mobile screens, but should work fine on laptops and desktops (maybe tablets, but I’m not testing that since it isn’t part of the assignment). Similarly, I will not explain any code relating to dynamic creation of HTML elements, as it is only relevant to the way the data is displayed and not how the simulation itself is run.

**Overview**

This program uses a Map as its primary data structure, which is a collection of key-value pairs. In this particular case, the keys are strings containing the names of the fish and the values are the corresponding count. For example, [“sea bass”, 96] is a key-value pair in the Map.

The data contained in the key-value pairs is displayed in HTML tables. There are four tables in total, though the first two tables are simply the same data ordered in different ways for the user’s reference:

**Table 01**: The original data I collected, sorted by prevalence

**Table 02**: The original data I collected, sorted alphabetically

**Table 03**: The results of the simulation

**Table 04**: The difference between the simulated data and the collected data

The data in Tables 03 and 04 is generated and displayed by clicking the “Run Simulation” button, which calls a function that randomly generates 500 fish using a weighted number generator derived from the original data. The difference table (Table 04) highlights the simulation’s accuracy by color-coding the degree of difference between simulated and collected data:

|  |  |  |  |
| --- | --- | --- | --- |
| **No difference** | **Under 1%** | **Between 1% and 2%** | **Over 2%** |

Figure 03: The percent difference between the simulated data and the collected data is denoted by color.

**The runSimulation() function**

The simulation itself is based around an array of 100 “fish,” called weightedArray in the code. The elements in weightedArray are added by iterating over the key-value pairs in a Map where the keys are strings containing the names of the fish and the values are integers representing their corresponding spawn rates. For example, the sea bass had a spawn rate of 19.20%, so the value for the key “sea bass” would be “19.”[[2]](#footnote-3)

// evaluate each key-value pair in fishProbMap

for (var [key,value] of fishProbMap){

for (var i = 0; i <= value; i++){

// add the key to weightedArray as many times as its value

// for example, if the key sea bass has a value of 19, it will be added to weightedArray 19 times

weightedArray.push(key);

}

}

This array is necessary for the weighted random generator function, which generates an integer between 0 and 99 and returns the fish at that index in weightedArray. The more times the fish is represented in the array, the higher the probability that the random number generator generates an index whose element is that fish.

The random generator function is called 500 times, and the result is stored in counter variables for each fish. For example, if the generator function returns an index whose element is “sea bass,” then a variable called seaBassCount will be incremented. The counter variable that is incremented is determined through a series of if-else statements.

After 500 generations, a new Map is created. This time, the values for each fish key are equal to the counter variable. For example, [“sea bass”, seaBassCounter] would be the key-value pair for the sea bass. The program then calls a function to display the simulation data in its own HTML table. The difference table is populated in a similar way, this time subtracting the values in the simulation from the values in the original table. A series of if-else statements determines color formatting for the table.

After the table is generated, the user can click the “run simulation” button again and new tables will be generated.

**Discussion**

I realized too late that I had not adequately account for the fact that some ocean fish *only* spawn near the pier, namely the mahi-mahi and great trevally. When I collected my data, I had walked along the entire beach and caught every fish that spawned. Since the pier only makes up a small area of the beach, and because there is no guarantee that a fish will even spawn there in the first place, the mahi-mahi and great trevally are greatly underrepresented in this data. Their underrepresentation leads to a permanent difference of -1 (-0.20%) in the difference table. If I were to do this project again, I would either narrow my location parameter to *only* the pier.

To take this model further, I would collect more data (thousands of data points instead of hundreds), which would greatly increase accuracy. It would also be interesting to collect data for every location, at every time of day, for every month of the year…but I don’t have that kind of time. I would also like to write a program that tracks the results of the simulation each time it is run so I could compare it to the weightedArray. Mathematically, I know the count should normalize toward the values in weightedArray in the long run, but I’d still like to prove it.

**Works Cited**

Greenwald, Will. “Animal Crossing: New Horizons (for Nintendo Switch) Review.” *PCMAG*, 2 Apr. 2020,

www.pcmag.com/reviews/animal-crossing-new-horizons-for-nintendo-switch.

Knight, Rosie. “ANIMAL CROSSING: NEW HORIZONS Smashes Digital Sales Records.” *Nerdist*, 29 Apr.

2020, nerdist.com/article/animal-crossing-new-horizons-breaks-sales-records/.

Lee, Julia. “Animal Crossing: New Horizons Fish Guide and Complete List.” *Polygon*, Polygon, 23 Mar.

2020, [www.polygon.com/animal-crossing-new-horizons-switch-acnh](http://www.polygon.com/animal-crossing-new-horizons-switch-acnh)

guide/2020/3/23/21190775/fish-locations-times-month-day-list-critterpedia.

1. Notice that the table contains non-fish like “tire” and “boot.” Just as in previous iterations of “Animal Crossing,” the latest version also features “junk” items that spawn as fish. The only thing more disappointing than a sea bass is a tire. [↑](#footnote-ref-2)
2. For fish with spawn rates under 1%, this resulted in a value of 0 for its key. This is the result of poor data collection that I will discuss in the Discussion section. [↑](#footnote-ref-3)