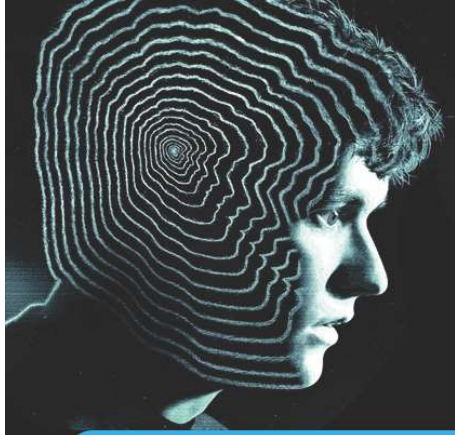
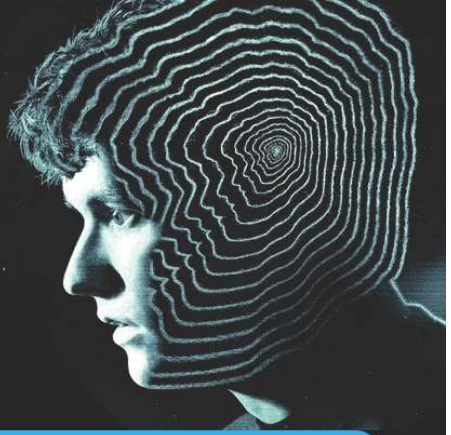


BLACK MIRROR

BANDERSNATCH

A SIMULATION



THE CHALLENGE

Given the recent release of Netflix's choose-your-own-adventure movie, Bandersnatch, I wanted to find out the probability that users will reach each different ending using Monte Carlo random simulations. In order to do so, I mapped out each decision that could be made by the viewer in Python and graphed my results using matplotlib.

the solution:
given that at each branch, the viewer is given a boolean option, I simply recreated the choices with a random number generator at each branch. There's usually a 50-50 chance for each option. However, some choices are contingent on choices previously made by the viewer, so those were more complex. Then, I recorded the frequency of each ending after each trial and visualized the resulting data in a graph after 10,000 trials.

As an added bonus, I also simulated the frequency at which viewers would discover a certain number of endings. So, for example, some viewers may only try one play-through and find one ending, others may find all 11. This also used Monte Carlo simulation.

each function is one branch

at each branch, a random boolean choice is made

some choices are saved

in order to keep track of choices that will have an impact later.

a final ending is returned

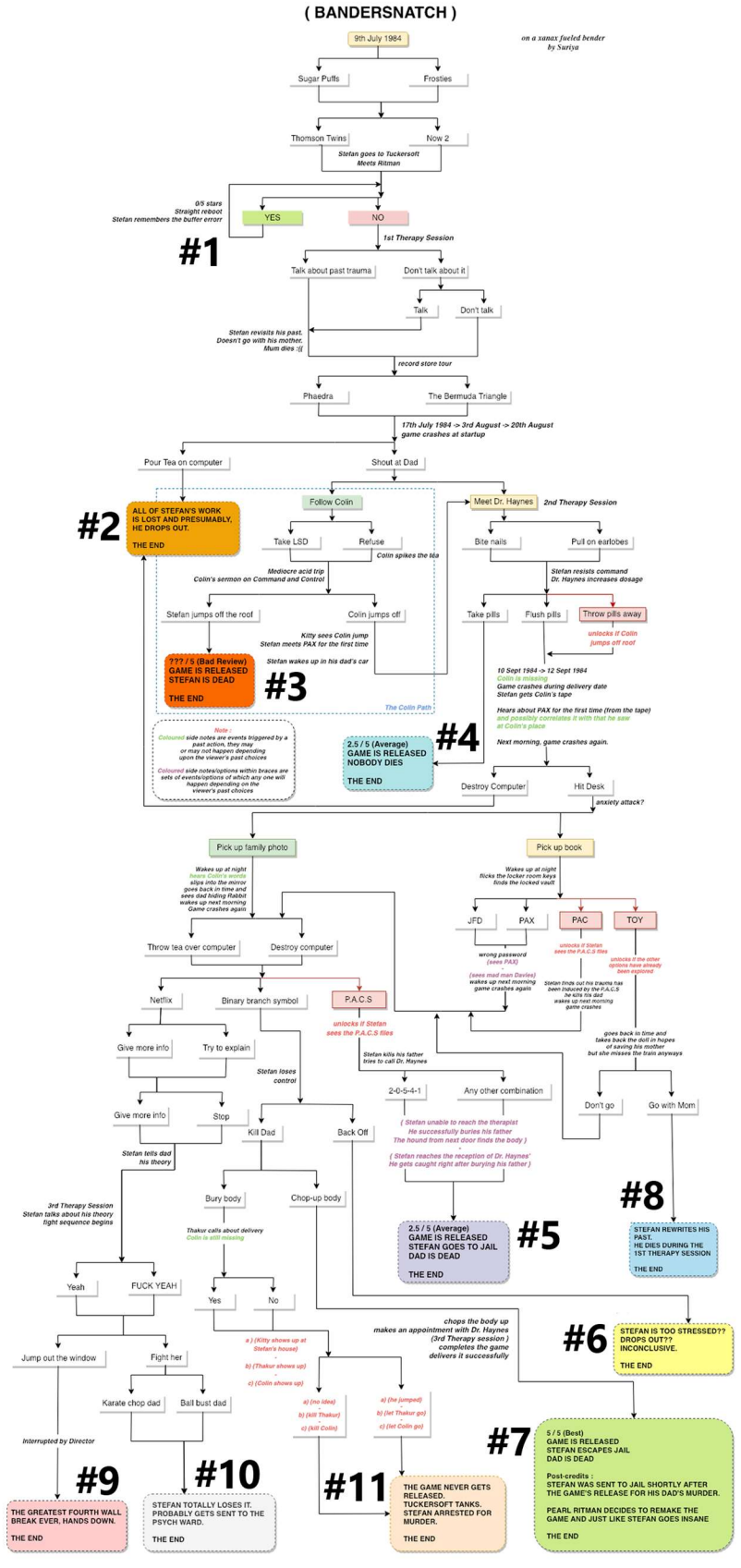
```
def colin(self):
    choice = random.randrange(0,2)
    self.stepcount += 1
    if choice == 0:
        result = 3
    else:
        self.colinDeath = True
        result = self.haynes()
    return result
def haynes(self):
    choice = random.randrange(0,2)
    self.stepcount +=1
    if choice == 0:
        result = self.noPills()
    else:
        result = 4
    return result
```

viewers are given a random retention rate

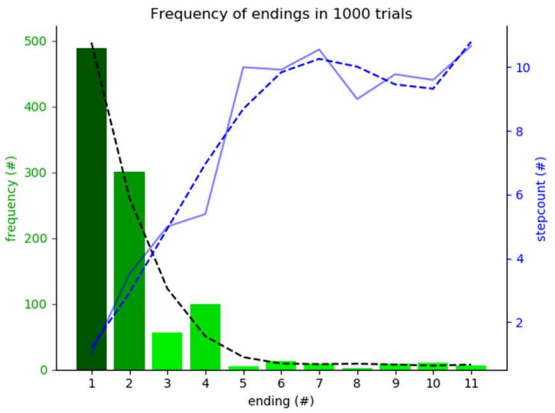
each viewer is assigned a random number of watch repetitions in order to randomize probability. The resulting endings are sorted and recorded.

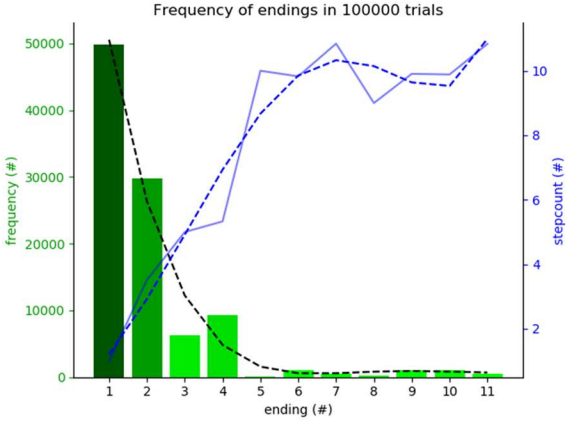
```
# repeats for n
for i in range(n):
    # selects a random retention rate fo
    t = random.randrange(1,limit)
    trial = Bandersnatch()
    results = trial.run(t)[0] # records
    results[:] = (value for value in res
    uniques = len(results) # uses list w
    # COUNTS HOW MANY ENDINGS DISCOVERED
    count[uniques-1] += 1

# bar graph color changes with value --
barcolor = [0 for i in range(11)]
for i in range(len(count)):
    barcolor[i] = [1-count[i]/(max(count
"""
STYLING SOURCES (in addition to in part
```

THE RESULTS

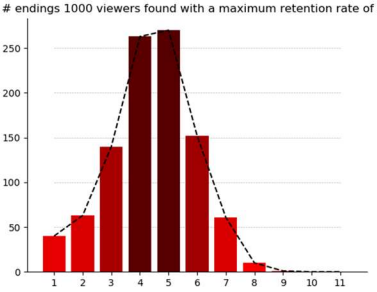


source: https://www.reddit.com/r/blackmirror/comments/aatz76/a_bandersnatch_flowchart_i_made_i_cant_even_go/

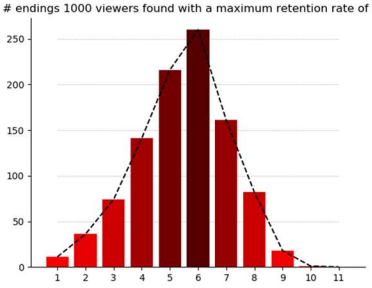




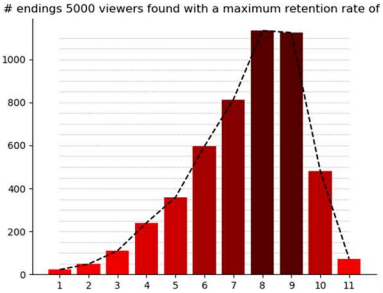
endings 1000 viewers found with a maximum retention rate of 50



endings 1000 viewers found with a maximum retention rate of 100



endings 5000 viewers found with a maximum retention rate of 300



ANALYSIS AND REFLECTION

According to the data, the fewer steps/branches it takes to reach an ending, the more often that ending is reached. This can be seen in the first two graphs (identical besides the number of trials) where stepcount is superimposed over frequency. Stepcount and frequency are seen to have an inverse relationship.

What's interesting is that endings #3 and #5 do not follow this trend. #5 can be easily explained because of its "secrecy" and abundance of pre-requisites to reach, but #3 seems a bit more peculiar.

Furthermore, as number of viewers increases and number of play-throughs per viewer increases, the number of endings discovered also increases. This is demonstrated in the red graphs below the two green ones.

want to try the code yourself? visit github.com/suddenlykevin/cs550datasim and run datasim.py and datasim2.py as instructed to generate your own results!

sources:
matplotlib styling - www.randalolson.com/2014/06/28/how-to-make-beautiful-data-visualizations-in-python-with-matplotlib/
matplotlib separate scales for y axes - https://matplotlib.org/examples/api/two_scales.html
label all bars in bar graph - https://stackoverflow.com/questions/26131822/how-to-display-all-label-values-in-matplotlib/
setting different colors for each bar - https://stackoverflow.com/questions/18973404/setting-different-bar-color-in-matplotlib-python
specifying layer order of graphs - https://stackoverflow.com/questions/37246941/specifying-the-order-of-matplotlib-layers
specifying when to run certain code - https://stackoverflow.com/questions/6523791/why-is-python-running-my-module-when-i-import-it-and-how-do-i-stop-it
smoothing out a curve - https://stackoverflow.com/questions/46633544/smoothing-out-curve-in-python/
divide one list by another - https://stackoverflow.com/questions/14434605/divide-one-list-by-another-list
remove certain values from a list - https://stackoverflow.com/questions/1157106/remove-all-occurrences-of-a-value-from-a-list