Introduction and Rationale

After reading the game rules, we determined the payoff matrix of the game (public goods withholding) to be :

Tribesman 1/ 2	Hunt	Slack
Hunt	0, 0	-3, 1
Slack	1, -3	-2, -2

Upon closer examination, the game appeared to resemble a modified ‘stag hunt’ scenario.

	Stag	Hare
Stag	A, a	C, b
Hare	B, c	D, d

Fig. 1: Generic stag hunt

Where B,b > A,a > D,d > C,c instead of the traditional A,a > B,b,D,d > C,c.

There are two nash equilibria, one where both players hunt (better payoff) and one where both players defect (better risk management).

By backward induction, we can conclude that both parties will always choose the dominant strategy of slacking (known as ‘hunting the hare’ in the classic stag hunt). Knowing that the opponent will choose slack (since a payoff of 1 is better than 0), we know that our best choice is to slack as well (since a payoff of -2 is better than -3). Of course, the public goods mechanic does help balance the game in favor of hunting.

To better our understanding of the game, we created a test suite (https://github.com/egordon/Brilliant-HungerGames-TestSuite) that we could use to test our code and generic algorithms. Overall, we found that slacking was the optimum strategy. However, with enough reputation-based strategies to ‘punish’ the slackers, the all-slack strategy became a liability. The key was to determine the significance of our reputation to our opponents (by correlating our reputation with our utility each round), allowing our algorithm to quickly detect when it could get away with slacking.

In General, Our Algorithm Is As Follows:

- We determine whether our reputation is correlated to our utility each round.
- If there is a positive correlation, then we want to hunt more often to increase our reputation and therefore our utility.
- If there is no correlation or a negative correlation, we slack, thus increasing our utility while lowering our reputation.

Step-By-Step Description

Technical Note: Our algorithm requires the “random” and “scipy” modules, which we import dynamically.

1. Add our current reputation to the end of the reputation history array. Pop the oldest value in the array.
2. Determine whether we are in the initial stage of the game (determined to be optimal at 25 rounds through testing). If...
    a. Yes, we gradually increase our likelihood of hunting to build a strong positive slope in our reputation, starting at a ~0% chance of hunting at round 1, and reaching 100% by the end of our defined early game.
    b. No, we set our likelihood of hunting equal to the correlation coefficient (of the statistical correlation test ‘Pearson’s r’) scaled by the respective correlation significance.
    c. **NOTE:** If the correlation coefficient is less or equal to 0, there is no chance of hunting, as a negative correlation explicitly favors slacking and a correlation of 0 determines that our opponents are not punishing us for slacking.
3. If we are not in the early game, enable "nice" edge detection.
    a. **"NICE" EDGE DETECTION:** Hunt with players with very high reputations (more than 90%), and Slack with players with very low reputations (less than 10%). Those algorithms are consistent enough to be trusted.
4. For each player, calculate a uniform random number from [0,1]. If that number is less than our current likelihood of hunting, we hunt.
5. Return the list of decisions.

6. Calculate our total utility for the round and add it to the Utility History Array.
7. Calculate the correlation coefficient and significance between with utility and reputation data from the past 25 rounds. This is used for next round's likelihood of hunting.