Supplementary Information

Title: Vaccinating behavior guided by imitation and aspiration

Author names and affiliations

Md. Rajib Arefin[a,b*], Tanaka Masaki[a], Jun Tanimoto[a,c]

[a]Interdisciplinary Graduate School of Engineering Sciences, Kyushu University, Kasuga-koen, Kasuga-shi, Fukuoka 816-8580, Japan

[b]Department of Mathematics, University of Dhaka, Dhaka-1000, Bangladesh

[c]Faculty of Engineering Sciences, Kyushu University, Kasuga-koen, Kasuga-shi, Fukuoka 816-8580, Japan

***Corresponding author**

arefin.math@du.ac.bd

This file includes the detailed derivation of Eq. (9) in the main text (section A). It also provides the source code (c++) of the numerical simulation (section B).

Section A: Derivation of Eq. (9)

Under the pure aspiration dynamic ($\alpha = 0$), the expression for steady-state frequency of vaccinators, as in Eq. (8),

$$x^*_{asp} = \frac{P^S_{NV}}{P^S_{NV}+P^S_V} = \frac{1+\exp[-k(\omega_V-\langle\pi_V\rangle)]}{2+\exp[-k(\omega_V-\langle\pi_V\rangle)]+\exp[-k(\omega_{NV}-\langle\pi_{NV}\rangle)]}.$$

Under weak selection $k << 1$, the above expression can be written as,

$$\begin{aligned} x^*_{asp} &\approx \frac{1+1-k(\omega_V-\langle\pi_V\rangle)}{2+1-k(\omega_V-\langle\pi_V\rangle)+1-k(\omega_{NV}-\langle\pi_{NV}\rangle)} \\ &= \frac{2-k(\omega_V-\langle\pi_V\rangle)}{4-k(\omega_V-\langle\pi_V\rangle)-k(\omega_{NV}-\langle\pi_{NV}\rangle)} \\ &= \frac{2-k(\omega_V-\langle\pi_V\rangle)}{4(1-\frac{k}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle))} \\ &= \frac{1}{4}\big(2-k(\omega_V-\langle\pi_V\rangle)\big)\left(1-\frac{k}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle)\right)^{-1}. \end{aligned} \quad \text{(A1)}$$

We presume that aspiration levels of vaccinators and non-vaccinators fall between the minimum and maximum value of their payoffs ($\langle\pi_V\rangle$ or $\langle\pi_{NV}\rangle$). More $k \ll 1$, we can perceive that, $\left|\frac{k}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle)\right| < 1$. Thus, we can use the binomial expansion for $\left(1-\frac{K}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle)\right)^{-1}$,avoiding higher order terms of k as,

$$\left(1-\frac{K}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle)\right)^{-1} \approx 1+\frac{k}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle). \quad \text{(A2)}$$

Using (A2) in (A1),

$$x^*_{asp} \approx \frac{1}{4}\big(2-k(\omega_V-\langle\pi_V\rangle)\big)\left(1+\frac{k}{4}(\omega_V+\omega_{NV}-\langle\pi_V\rangle-\langle\pi_{NV}\rangle)\right) \quad \text{(A3)}$$

Simplifying the above expression and avoiding terms that involve k^2, one can finally obtain Eq. (9) as in the main text,

$$x^*_{asp} \approx \frac{1}{2}+\frac{k}{8}[(\omega_{NV}-\omega_V)+(\langle\pi_V\rangle-\langle\pi_{NV}\rangle)] \quad (9)$$

Section B: Source code for the numerical simulation

Below we provide the source code (c++) of our numerical simulation. This code especially generates data for Fig. 3(a, b) in the body text. All other figures can be generated by tuning relevant parameters.

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <list>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <sstream>
#include <string>
#include <cstdio>
#include <time.h>
#include <iomanip>
using namespace std;
using std::setw;

int main()
{


# define times 300
# define GENERATION 20000
# define K 10// loop



        string file1 = "fes.csv";
        ofstream Data1(file1);
        Data1 << "Cr,e,FES" << endl;

        string file2 = "vc.csv";
        ofstream Data2(file2);
        Data2 << "Cr,e,VC" << endl;


        string file3 = "w_v.csv";
        ofstream Data3(file3);
        Data3 << "cr,e,wv" << endl;

        string file4 = "w_nv.csv";
        ofstream Data4(file4);
        Data4 << "Cr,e,wnv" << endl;

        string file5 = "alpha.csv";
        ofstream Data5(file5);
        Data5 << "Cr,e,alpha" << endl;


double pi_V, pi_NV, w_v[GENERATION + 1], w_nv[GENERATION + 1], R_inf[times + 1];

double alp[GENERATION + 1], x[GENERATION + 1], alpha;
```

```
	double R0 = 2.5; // Basic reproduction number

	for (int cost = 0; cost < 101; cost++)  // loop for vaccination cost
	{
		double Cr = cost / 100.0;

		for (int ef = 0; ef < 101; ef++) // loop for efficacy
		{

			double e = ef / 100.0; w_nv[0] = -0.5; w_v[0] = -0.5;
			 alp[0] = 0.2; x[0] = 0.5;
			for (int j = 0; j < GENERATION; j++) // loop for repeated
seasons
			{
				R_inf[0] = 0.5;
				for (int i = 0; i < times; i++)
				{
					R_inf[i + 1] = (1 - e * x[j]) * (1 - exp(-R0 *
R_inf[i])); // estimating final epidemic size
				}

				double hv = x[j] * (e + (1 - e) * exp(-R0 *
R_inf[times])); //healthy and vaccinated
				double iv = x[j] * (1 - e) * (1 - exp(-R0 *
R_inf[times])); //infected and vaccinated
				double sfr = (1 - x[j]) * exp(-R0 * R_inf[times]);
//successful free rider
				double ffr = (1 - x[j]) * (1 - exp(-R0 *
R_inf[times])); //failed free rider
				pi_V = (-Cr * hv - (Cr + 1) * iv) / x[j]; //average
payoff of vaccinators
				pi_NV = -ffr / (1 - x[j]); //average payoff of non-
vaccinators

			  //assigning alpha and aspiration levels
			alpha = 0.5; double omega1 = -0.5; double omega2 = -0.2;
// omega1 and omega2 represent aspiration levels for vaccinators and non-
vaccinators

				double p_V_NV = x[j] * (1 - x[j]) * 1 / (1 + exp(-
(pi_NV - pi_V) * K)); //imitative probability from V to NV
				double p_V_C = 1.0 / (1 + exp(-(omega1 - pi_V) * K));
//Aspiration-led switching probability for vaccinators
				double p_NV_V = x[j] * (1 - x[j]) * 1.0 / (1 + exp(-
(pi_V - pi_NV) * K)); //imitative probability from NV to V

				double p_NV_C = 1.0 / (1 + exp(-(omega2 - pi_NV) *
K)); //Aspiration-led switching probability for non-vaccinators
				x[j + 1] = x[j] + 0.1 * (p_NV_V - p_V_NV);
				double x_imt = x[j + 1];
				x[j + 1] = x[j] + 0.1 * ((1 - x[j]) * p_NV_C - x[j] *
p_V_C);
				double x_aspp = x[j + 1];
				x[j + 1] = alpha * x_imt + (1 - alpha) * x_aspp;

// In case of time dependent ‘alpha’ and ‘aspirations’, one can use following
expressions, and then substitute ‘alpha = alp[j]’, ‘omega1 = w_v[j]’, and ‘omega2
= w_nv[j]’ in the line with comment “//assigning alpha and aspiration levels”
			  //double h = 0.2;  // learning rate
```

```
                        //w_v[j + 1] = (1-h) * w_v[j] + h * pi_V;
                        //w_nv[j + 1] = (1-h) * w_nv[j] + h * pi_NV;
                        //alp[j + 1] = alp[j] + 1.0 * alp[j] * (1 - alp[j]) *
(x_imt - x_aspp);  // updating alpha

                    }
                    Data1 << Cr << "," << e << "," << R_inf[times] << endl;
                    Data2 << Cr << "," << e << "," << x[GENERATION] << endl;
                    Data3 << Cr << "," << e << "," << w_v[GENERATION] << endl;
                    Data4 << Cr << "," << e << "," << w_nv[GENERATION] << endl;
                    Data5 << Cr << "," << e << "," << alp[GENERATION] << endl;
            }
        }

        Data1.close();
        Data2.close();
        Data3.close();
        Data4.close();
        Data5.close();
}
```