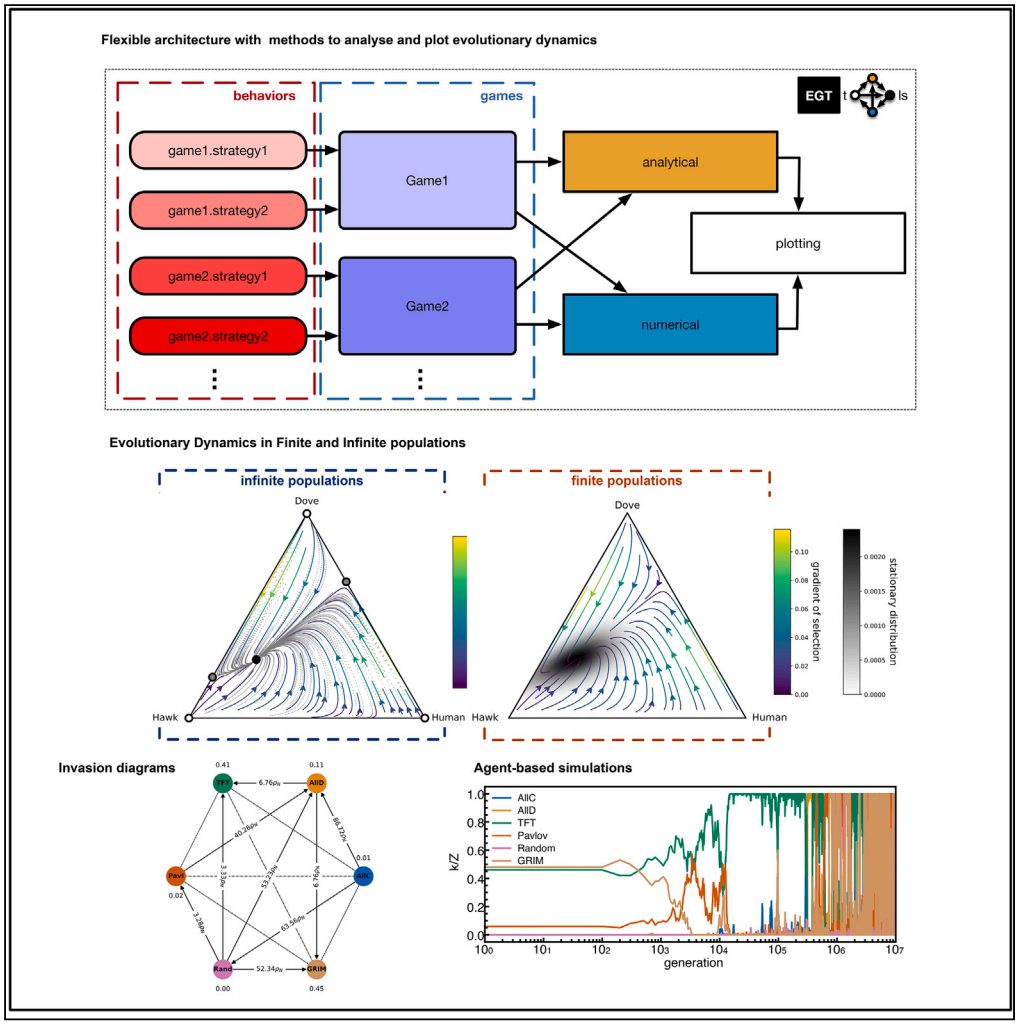
iScience

CellPress
OPEN ACCESS

Article

EGTtools: Evolutionary game dynamics in Python



Elias Fernández Domingos, Francisco C. Santos, Tom Lenaerts

elias.fernandez.domingos@ulb.be

Highlights

Evolutionary Game Theory (EGT) provides a framework to study collective behavior

EGTtools provides fast implementations of analytical and numerical EGT methods

EGTtools implements methods to analyze finite and infinite populations

We illustrate the use of the library with concrete examples

Fernández Domingos et al., iScience 26, 106419
April 21, 2023
https://doi.org/10.1016/j.isci.2023.106419

iScience

CellPress
OPEN ACCESS

Article

EGTtools: Evolutionary game dynamics in Python

Elias Fernández Domingos,[1,2,5,*] Francisco C. Santos,[3] and Tom Lenaerts[1,2,4]

SUMMARY

Evolutionary Game Theory (EGT) provides an important framework to study collective behavior. It combines ideas from evolutionary biology and population dynamics with the game theoretical modeling of strategic interactions. Its importance is highlighted by the numerous high level publications that have enriched different fields, ranging from biology to social sciences, in many decades. Nevertheless, there has been no open source library that provided easy, and efficient, access to these methods and models. Here, we introduce EGTtools, an efficient hybrid C++/Python library which provides fast implementations of both analytical and numerical EGT methods. EGTtools is able to analytically evaluate a system based on the replicator dynamics. It is also able to evaluate any EGT problem resorting to finite populations and large-scale Markov processes. Finally, it resorts to C++ and MonteCarlo simulations to estimate many important indicators, such as stationary or strategy distributions. We illustrate all these methodologies with concrete examples and analysis.

INTRODUCTION

Classical game theory[1] typically analyzes interactions between two (or a few) players and aims to answer how each individual can maximize their utility. Such question, from a mathematical perspective, becomes very intricate and complex as a player must consider the utilities of all other players and any possible set of beliefs. Thus, following the famous work of Nash,[2] it is often assumed that all players act in a way that maximizes their utility and believe that others will do the same. This simplifies the analysis and transforms it into a search for the points of equilibrium at which no player has any incentive to change its strategy (Nash equilibrium). This assumption requires that players have perfect knowledge about the game/environment and is referred to as rationality. Yet, it is often cumbersome to assume that individuals are rational in many social and biological systems, even in simple pairwise interactions. Moreover, whenever the problem requires a proper understanding of conflicts occurring in large populations, it becomes necessary to characterize the choices and strategies of many individuals throughout time, and not only at equilibrium. As such, in many real-world scenarios where game theory is applied, the goal is shifted toward the understanding of the complex ecologies of behaviors emerging from a given dilemma (or "game"). This is where evolutionary game theory (EGT) shines as a theoretical and computational framework.[3,4]

In EGT, the individual payoffs collected through strategic interactions in a given game are associated with a personal fitness. In a biological setting, the fittest individuals are more likely to produce offspring, thus, passing on their traits to the next generation. Hence, fitness has a direct interpretation in terms of relative death and birth rates. When applying this evolutionary concept to social systems, the most successful are more likely to be imitated. As a result, their strategies will have higher chances of being adopted by other members of the population in subsequent generations. Fitness therefore encodes a measure of social success. Overall, EGT allows for a convenient (formal and dynamical) similarity between social learning and Darwinian evolution, further broadening the spectrum of possible applications of evolutionary games.

Nevertheless, EGT models are complex, reflecting the sophistication of the interactions, the contexts being modeled, and the questions being addressed. This complexity may emerge from different factors. To name a few, individuals may portray different adaptation mechanisms,[5,6] roles, strategies, and tags[7,8]; when playing, they can choose from a finite number of strategies or sample a value from a continuous domain[9]; individuals may have the opportunity to spread signals or information about others (e.g., reputations)[10–16] populations can show a (static or dynamic) interaction structure.[17–22] The sources of complexity are endless.

[1]Machine Learning Group, ULB, Campus la Plaine, Brussels 1050, Belgium

[2]AI Lab, VUB, Pleinlaan 9, Brussels 1050, Belgium

[3]INESC-ID & Instituto Superior Técnico, Universidade de Lisboa, IST-Taguspark, Porto Salvo 2744-016, Portugal

[4]Center for Human-compatible AI, UC Berkeley, 2121 Berkeley Way, Berkeley, CA 94720, USA

[5]Lead contact

*Correspondence: elias.fernandez.domingos@ulb.be

https://doi.org/10.1016/j.isci.2023.106419

Figure 1. Simplex S_2 for the prisoner's dilemma

The values along the line represent the frequency of cooperators in the population. The arrow indicates the direction of the gradient of selection. The circles represent rest or equilibrium points. The black circle represent a stable equilibrium and the white an unstable one. In this example, all members of the population adopting strategy D is the only stable equilibrium.

As a result, the study of evolutionary dynamics increasingly relies on computational methods, as more and more questions range outside analytical tractability.[23] However, large-scale computer models, often grounded on agent-based simulations, can easily become black boxes, possibly harder to understand than the actual real-world system we want to explain. Therefore, agent-based models should be built such that, within reasonable limits, they can be compared with analytical results. For this reason, EGT models often combine large-scale simulations with formal approximations.

Despite the importance of creating a common framework to build and analyze EGT models, both from computational and analytical perspectives, no standard library provides an easy, efficient, and accessible route to these methods and models. Here, we introduce EGTtools[24] STAR Methods, a hybrid C++/Python library, which offers fast and parallel implementations of analytical, numerical and Monte-Carlo methods necessary to estimate efficiently the essential outputs of many EGT models. This library aspires to provide a standardized way for researchers to access and share EGT models and methods. In addition, the Python interface is aimed at facilitating the access to these, often complex, mathematical formulations to a larger scientific field.

EGTtools relies on a synergy between computational and analytical approaches, all implemented within the same framework. It provides the means to address a problem from a purely computational perspective (through large-scale agent-based simulations) as well as several types of analytical descriptions of the same models. Particularly, EGTtools allows for (1) large-scale multi-agent simulations and analytical descriptions that can either involve (2) the use of Markov Processes which closely follows the agent-based simulations, (3) Small-scale Markov Processes that approximate in a tractable way the same stochastic simulations, and (4) classic deterministic analysis obtained through differential equations (i.e., replicator equation). To illustrate the use of EGTtools in all these scenarios, we provide six detailed examples.

In the following, we introduce evolutionary games in large populations through replicator equation. Subsequently, we look into methods capable of addressing the stochastic effects and population dynamics in finite populations. Later, we discuss the scalability issues that may appear when analyzing populations which can adopt a large number of strategies and how we can overcome them either with analytical approximations or dimensionality reduction. Finally, we discuss how all these perspectives can be complemented with large-scale numerical simulations. Examples accompany each section on how to use EGTtools to apply these methods and models. Finally, we compare EGTtools to other existing libraries and discuss their advantages.

Evolutionary games in large populations

Let us consider a well-mixed population of size Z in which individuals engage in a Game (or social dilemma). Each individual i may adopt any strategy e_i of the n_s available strategies.

When interacting in a well-mixed population, the probability of interacting with any other individual is equal. Each interaction is ruled by a game, e.g., a prisoner's dilemma (PD). The expected payoff of the game obtained through (one or) many interactions is transformed into individual fitness. Such fitness defines the probability that an individual will produce offspring in the next generation. In such Darwinian dynamics, the frequency of a strategy i, x_i, grows in proportion to its fitness. Thus, we may consider a population state as the tuple $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_{n_s}) \in \mathbb{R}^{n_s}$ representing the frequency (or proportion) of adoption of each strategy by individuals in the population.[3]

The possible states (or mixed strategies) of the population and their gradient can be represented in a simplex S_{n_s}. Figure 1 shows the simplex S_2 representing all possible population states in a PD.[4] The arrow indicates the tendency of a strategy to grow, and the circles represent fixed points or equilibria in the game.

The replicator equation represents the dynamics of competing individuals in an infinite population ($Z \to \infty$). It defines the rate at which the frequency of strategies in a population will change (the gradient of selection).[25] It is often found in the form of Equation 1,[3] where x_i represents the frequency of strategy i in the population, and $f_i(\boldsymbol{x}) = \Pi_i(\boldsymbol{x})$ its fitness (or expected payoff), given the current population state $\boldsymbol{x}$. The term $\sum_{j=1}^{N} x_j f_j(\boldsymbol{x})$ represents the average fitness of the population in state $\boldsymbol{x}$.

$$\dot{x}_i = x_i \left(f_i(\boldsymbol{x}) - \sum_{j=1}^{N} x_j f_j(\boldsymbol{x}) \right) \quad \text{(Equation 1)}$$

The previous differential equation represents a selection dynamic ruled by reproductive success. However, it is common to adopt a different mechanism when looking at human social interactions in economic games. Here, changes in the population throughout generations are a product of *social learning*, by which individuals imitate each other as a function of their fitness difference.

Concretely, we assume that at a given moment in time, a randomly sampled individual j seeks a role model in the population by randomly sampling another individual i. If the fitness of i is higher than that of j, then the latter imitates the former with a probability proportional to the fitness difference $f_{ij}(\boldsymbol{x})$ and the growth rate is defined by the differential Equation 2.[3]

$$\dot{x}_i = x_i \sum_j \left(f_{ij}(\boldsymbol{x}) - f_{ji}(\boldsymbol{x}) \right) x_j \quad \text{(Equation 2)}$$

The fitness of each individual $f_i(\boldsymbol{x})$ is represented by the expected payoff $\Pi_i(\boldsymbol{x})$. Therefore, the fitness difference can be represented as $f_{ij}(\boldsymbol{x}) = [f_i(\boldsymbol{x}) - f_j(\boldsymbol{x})]_+ = [\Pi_i(\boldsymbol{x}) - \Pi_j(\boldsymbol{x})]_+$, which is only defined for positive values, i.e., when $f_i(\boldsymbol{x}) > f_j(\boldsymbol{x})$. Here, we also assume that the relationship between imitation probability and fitness difference is linear and 1 : 1. This is not always the case, and often an extra parameter β is included, so that the importance of the difference in payoffs can be adapted to each context. Nevertheless, assuming this direct relationship, $f_{ij}(\boldsymbol{x}) - f_{ji}(\boldsymbol{x}) = [\Pi_i(\boldsymbol{x}) - \Pi_j(\boldsymbol{x})]$. If we replace this in Equation 2 we obtain an analogous equation to 1. Finally, making some substitutions we can write:

$$\dot{x}_i = x_i(\Pi_i(\boldsymbol{x}) - \overline{\Pi}(\boldsymbol{x})) \quad \text{(Equation 3)}$$

where $\overline{\Pi}(\boldsymbol{x}) = \sum_{j=1}^{N} x_j f_j(\boldsymbol{x})$ represents the average fitness of the population. This differential equation completely characterizes the strategical variations in the population by representing the direction of change for each strategy. If $\dot{x}_i > 0$, strategy i will grow. Contrarily, if $\dot{x}_i < 0$ it will decrease. An equilibrium is achieved whenever $\dot{x}_i = 0$ (the gradient of change in population state is 0), and such an equilibrium may be stable or unstable. An equilibrium x^* is said to be evolutionary stable if an infinitesimal change in the frequencies of strategies in the population produces a gradient toward x^*, whereas it is unstable if the gradient moves away from x^*. A visual example can be found in Figure 1, which represents the S_2 simplex for the PD. The gradient for strategy C, $\dot{x}_C < 0$, is strictly decreasing unless the whole population adopts C. Therefore, when the population is in $x_C^* = 1$ (all individuals adopt C), a change in a single player's strategy would move the population toward $x_C^* = 0$ (all individuals adopt D). We then say that the game has a stable equilibrium at $x_C^* = 0$.

This Evolutionary Stable State (ESS)[26,27] provides a more strict stability concept than the well known Nash equilibrium.[2] A stable equilibrium here not only indicates a situation from which no player can benefit by deviating unilaterally, but also that a population in an ESS is robust to small changes because of noise, i.e., a mutant strategy will not drive the population to a different state, instead, the population will return to the ESS.

Examples 1 and 2 (and Figure 2) show how to use EGTtools to visualize the dynamics of a Hawk-Dove game with two or three strategies respectively. The same concept presented in these examples can be applied to any other game, all that is required is to define a payoff matrix containing the payoffs of each strategy against any other.

Population-based Markov processes

The limit of infinite populations is convenient from a technical point of view, as it allows the use of relatively simple differential equations to model complex evolutionary processes. However, in many situations, when modeling realistic problems, we cannot neglect the stochastic effects that come along when individuals interact in finite populations.[5] This is the case of many collective endeavors, from animal group hunting

Example 1. Replicator equation: population with two strategies

Below you can find an example of how EGTtools can be used to visualize the gradient of selection for a 2-strategy Hawk-Dove game (the results are shown in Figure 2A). In a Hawk-Dove game, the Hawk strategy represents an aggressive individual, i.e., when it encounters other individuals (of any strategy) it will choose to fight. Doves, however, prefer to avoid confrontation. For this reason, when Hawks encounter other Hawks, they will fight and have a chance of ending badly injured. We represent this with a payoff of −0.5. On the other hand, when a Hawk encounters a Dove, the Dove will escape and the Hawk will be able to keep the resource in dispute. We represent this with a payoff of 2 for the Hawk and of 0 for the Dove. Finally, when Doves meet, they will share the resource, but this will give them no surplus and they will get a payoff of 0.

The example shows how to obtain an array of gradients (`calculate_gradients`), extract the roots (`find_roots`) and check their stability (`check_replicator_stability_pairwise_games`), and plot them (`plot_gradients`)

The example below shows how to obtain an array of gradients, extract the roots and their stability and plot them. The same process can be applied to any other 2-strategy 2-player game.

```
import numpy as np
import egttools as egt
from egttools.analytical.utils import (calculate_gradients, find_roots, check_replicator_
  stability_pairwise_games, )
# Calculate gradient
payoffs = np.array([[-0.5, 2.], [0., 0]])
x = np.linspace(0, 1, num = 101, dtype = np.float64)
gradient_function = lambda x: egt.analytical.replicator_equation(x, payoffs)
gradients = calculate_gradients(np.array((x, 1 - x)).T, gradient_function)
# Find roots and stability
roots = find_roots(gradient_function, nb_strategies = 2, nb_initial_random_points = 10,
  method = ''hybr'')
stability = check_replicator_stability_pairwise_games(roots, payoffs)
# Plot the gradient
egt.plotting.plot_gradients(gradients[:, 0], xlabel = ''frequency of hawks'', roots =
  roots, stability = stability)
```

and warfare,[28–30] to numerous human affairs, such as small community collective projects, macroeconomic relations and the famous world summits on climate change,[31–36] where group and population sizes are often comparable and of the order of the hundreds.

For such population sizes, stochastic effects play an important role and the behavioral dynamics is best described in terms of discrete birth-death type processes, leading to a finite population gradient of selection $G(k/Z)$[37], defined as the difference between the probabilities of increasing and decreasing the number of a given strategy by one, together with the stationary distribution of the Markov chain associated which characterizes the pervasiveness in time of a given composition of the population. Stochastic effects are amplified in the presence of errors of different sorts (inducing behavioral "mutations"), including errors of imitation.[5,23,38]

We now consider a finite population of Z individuals, who interact in groups of size $N \in [2,Z]$, in which they engage in strategic interactions (or games). Each individual can adopt one of the n_s strategies. The success (or fitness) of an individual can be computed as the expected payoff of the game in a given state $\boldsymbol{x}$.

We adopt a stochastic birth-death process combined with the pair-wise comparison rule[5,37,38] to describe the social learning dynamics of each of the strategies in a finite population (see Figure 3). At each time-step, a randomly chosen individual j adopting strategy $\boldsymbol{e_j}$ has the opportunity to revise her strategy by imitating (or not) the strategy of a randomly selected member of the population i. The imitation will occur with a probability p which increases with the fitness difference between j and (HTML translation failed). Here we adopt the Fermi function (see Equation 4), which originates from statistical physics and provides a well defined mapping between $\mathbb{R}^+ \rightarrow [0,1]$. Please also note that because the population is finite, instead of assuming the frequencies of each strategy in the population (x_i) we use the absolute value k_i so that $x_i \equiv [k_i/Z]$.

$$p \equiv \left[1 + e^{\beta\left(f_i(k_i) - f_j\left(k_j\right)\right)}\right]^{-1} \qquad \text{(Equation 4)}$$

Example 2. Replicator equation: population with three strategies

Below you can find an example on how to plot a vector field in a simplex S_3 (a triangle). Here we extend the previous example of the Hawk-Dove game to include an extra strategy, humans (the result can be seen in Figure 2B). Humans, like to feed the Doves, so when encountering them, they get a payoff of 1, whereas the Dove will get a payoff of 2 for the food. Yet, Humans will avoid the Hawks, and both Hawks and Humans will obtain a payoff of 0 when encountering each other. Finally, we assume that a Human does not get any benefit from encountering another Human (so receives a payoff of 0).

The example can be divided in two parts: (1) calculate the gradients and roots (through the helper function `plot_replicator_dynamics_in_simplex`), and (2) plot the simplex and the gradients inside it (done through the `Simplex2D` object returned by `plot_replicator_dynamics_in_simplex`. The `Simplex2D` class has several methods which can be used to customize the plot. In this example, we plot a triangle (to represent the simplex S_3), add labels for the strategies, draw the roots of the system and their stability, the gradients (and a color bar which will represent the magnitude of the gradient), and draw some trajectories inside the simplex, which start from random points. Drawing these trajectories can help the reader better understand the speed of convergence toward stable points.

```
import numpy as np
import matplotlib.pyplot as plt
from egttools.plotting.simplified import plot_replicator_dynamics_in_simplex
payoffs = np.array([[-0.5, 2, 0], [0, 0, 2], [0, 1, 0]]) type_labels = ['Hawk', 'Dove', 'Human']
# Create figure
fig, ax = plt.subplots(figsize=(10,8))
# calculate the gradients and roots
simplex, gradient_function, roots, roots_xy, stability = plot_replicator_dynamics_in_
  simplex(A, ax = ax)
plot = (simplex.draw_triangle()
    .add_vertex_labels(type_labels, epsilon_bottom = 0.1)
    .draw_stationary_points(roots_xy, stability)
    .draw_gradients(zorder = 0)
    .add_colorbar()
    .draw_scatter_shadow(gradient_function, 100, color = 'gray', marker = '.', s = 0.1))
ax.axis('off')
ax.set_aspect('equal')
plt.xlim((-0.05,1.05))
plt.ylim((-0.02, simplex.top_corner +0.05))
```

In Equation 4, $f_j(f_i)$ is the fitness of individual $j(i)$ and β, also known as inverse temperature, controls the intensity of selection and the accuracy of the imitation process. For $\beta \to 0$, individual fitness is but a small perturbation to random drift; for $\beta \to \infty$ imitation becomes increasingly deterministic. Also, k_i represents again the total number of individuals adopting strategy i. In addition, we consider that with a mutation (or exploration) probability μ, individuals adopt a randomly chosen strategy, freely exploring the strategy space. Overall, this adaptive process defines a large-scale Markov chain, in which the transition probabilities between states are defined in function of the fitness of the strategies in the population and their frequency. The complete characterization of this process becomes unfeasible as the number of possible population configurations scales with the population size and the number of strategies following $\binom{Z+n_s-1}{n_s-1}$.[39] Thus, for larger strategy spaces, computer simulations for the estimation of the model's parameters are often a requirement. For this reason, we first assume that $n_s = 2$ by taking as an example the PD introduced earlier, and, in the next section, we show that we can apply the case shown here for two strategies, to any number of them if we assume $\mu \to 0$. The probability that the number k of participants adopting a cooperative strategy C would increase $(T^+(k))$ or decrease $(T^-(k))$ can be specified as[5]:

$$T^+ = (1-\mu)\frac{Z-k}{Z}\frac{k}{Z-1}\left[1+e^{-\beta(f_C-f_D)}\right]^{-1} + \mu\frac{Z-k}{Z}$$
$$T^- = (1-\mu)\frac{k}{Z}\frac{Z-k}{Z-1}\left[1+e^{\beta(f_C-f_D)}\right]^{-1} + \mu\frac{k}{Z} \quad \text{(Equation 5)}$$

where $\mu\frac{Z-k}{Z}$ represents the probability of a mutation occurring and picking strategy D, and $(1-\mu)\frac{Z-k}{Z}\frac{k}{Z-1}$ the probability that instead, the imitation process occurs, and individuals adopting different strategies (in this case C and D) are selected. Therefore, the transition matrix T that fully characterizes the dynamics of

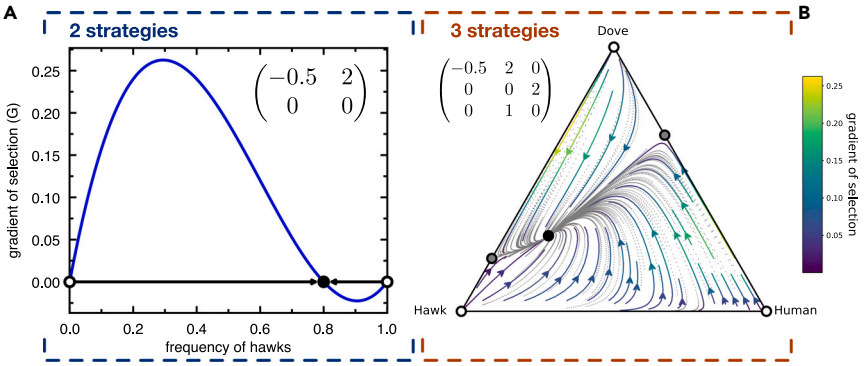


Figure 2. Gradient of selection of the replicator dynamics for a 2 and 3 strategy Hawk-Dove game

A white circle indicates an unstable equilibrium, a black circle indicates a stable one, and a gray circle indicates a saddle point. Panel (A) shows the gradient of selection for a two-strategy Hawk-Dove game. The plot also shows the direction of the gradient, the stationary points of the system and their stability. All this information can be easily plot with the *plot_gradients* function in EGTtools (see Example 1). Panel (B) shows similar information for the three-strategy version of the Hawk-Dove game. In this case, the gradients are plot over a simplex S_3 (a triangle) and their intensity is shown using colors.

the stochastic process consists of a mapping from each state k (the state in this case is defined by the number of cooperators) to an adjacent state k' as described by Equation 6.[3,36]

$$\begin{aligned} T_{i+1,i} &= T^- \\ T_{i,i+1} &= T^+ \\ T_{1,i} &= 1 - T^+ - T^- \end{aligned} \qquad \text{(Equation 6)}$$

Finally, the stationary distribution of the system p_e, i.e., the time the population spends at each state, can be computed analytically as the normalized eigenvector associated with the eigenvalue 1 of the transition matrix T of the Markov Chain.[38,40,41]

As occurs in the replicator dynamics, the tendency of the population to move toward certain states can be determined by a gradient, which is now discrete, denominated gradient of selection $G(k) = T^+(k) - T^-(k)$[37]. Therefore, if $G(k) < 0$ the number of cooperators in the population will tend to decrease. Similarly, the points where $G(k) = 0$ constitute rest points (or equilibria) of the system.

The imitation dynamics of social learning described here give us a powerful tool for the study of collective action. A vast literature has applied these models successfully to explain social interactions in public good games (PGGs),[42–46] threshold PGGs and collective-risk dilemmas (CRDs)[25,35–37,47–52] and other social dilemmas.[6,20,25,53–55] Moreover, a number of research works successfully explain experimental observations of collective behavior[16,56–59] through evolutionary models.

This stochastic process with pair-wise imitation is implemented in the *PairwiseComparison* class inside the *analytical* module of EGTtools. Examples 3 and 4 show how to analyze the same games as in Examples 1

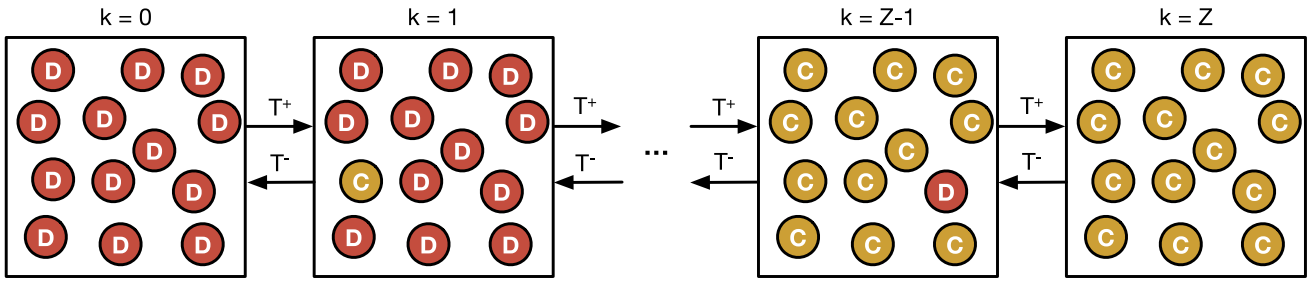


Figure 3. Stochastic birth-death process in a population of two strategies: cooperate and defect

The population is monomorphic when there are only defectors ($k = 0$) or only cooperators ($k = Z$). In these cases $T^+ > 0$ only if $\mu > 0$, i.e., a mutant has to appear so that the system may change state. In all other cases the transition probabilities (T^+ and T^-) depend on the payoff difference between defectors and cooperators, which is also a function of the population state.

Example 3. Stochastic population dynamics: population with two strategies

To model the stochastic effects that can appear in finite populations, EGTtools implements in the *PairwiseComparison* class a stochastic birth-death process in which individuals adapt through imitation (social learning). Moreover, changes in the population are produced through pairwise-comparison, where two individuals are randomly sampled, and the first individual will imitate the second with a probability proportional to their payoff difference (see section on population-based Markov processes).

The example below uses the same payoff structure of Example 1, i.e., we are looking again at a Hawk-Dove game. The result of this code is shown in Figure 4A.

```
import numpy as np
import egttools as egt
# Parameters
payoffs = np.array([[-0.5, 2.], [0., 0]])
nb_strategies = 2; Z = 100; beta = 1
pop_states = np.arange(0, Z + 1, 1)
# Define a game object that will hold the payoff matrix
game = egt.games.Matrix2PlayerGameHolder(nb_strategies, payoff_matrix = payoffs)
# Evolver and calculate gradients
evolver = egt.analytical.PairwiseComparison(Z, game)
gradients = np.array([evolver.calculate_gradient_of_selection(beta, np.array([x, Z-x]))
  for x in range(Z + 1)])
# Plot the results
egt.plotting.indicators.plot_gradients(gradients[:, 0], figsize=(6,5), marker_facecolor =
  'white', xlabel = ''frequency of hawks (k/Z)'', marker = ''o'', marker_size = 30, marker_
  plot_freq = 2)
```

and 2, however we now assume that we have a finite population of $Z = 100$ individuals. Although the results are similar, the stationary distribution in Figure 4B shows that the system will no longer end in the stable equilibrium deterministically, but rather, there is a probability distribution centered around such equilibrium.

Approximations and dimensionality reduction

The Markov chain described above can quickly become too complex for analytical description as the number of strategies, even for small population sizes (there are $\binom{Z+n_s-1}{n_s-1}$ possible population configurations). However, whenever in the limit where mutations are rare ($\mu \to 0$) it is possible to approximate the complete stochastic process by a Markov chain with a number of states given by the number of strategies. In this small mutation limit (SML),[38,40] when a new strategy appears through mutation, one of two outcomes occurs long before the occurrence of a new mutation: either the population faces the fixation of a newly introduced strategy, or the mutant strategy goes extinct.

Hence, there will be a maximum of two strategies present simultaneously in the population.[38,40] This allows us to describe the behavioral dynamics in terms of a reduced Markov chain of size n_s, whose transitions are defined by the fixation probabilities ρ_{ji} of a single mutant with strategy i in a population of individuals adopting another strategy j according to Equation 7.[5,41,60] Thus, fixation probabilities also characterize the evolutionary resilience of a strategy over another, and the increase (decrease) probabilities, $T^-(k)(T^+(k))$, are defined as indicated in Equation 8.

$$\rho_{ji} = \left(1 + \sum_{m=1}^{Z-1} \prod_{k=1}^{m} \frac{T^-(k)}{T^+(k)}\right)^{-1} \quad \text{(Equation 7)}$$

$$T^{\pm}(k) = \frac{Z-k}{Z}\frac{k}{Z-1}\left[1 + e^{\mp\beta\left(f_i - f_j\right)}\right]^{-1} \quad \text{(Equation 8)}$$

Also, the transition matrix T can now be constructed so that each entry represents the transition between monomorphic states, i.e., population states in which all individuals adopt the same strategy. Thus, the transition probability from strategy i to strategy j is $T_{i,j} = \rho_{ij}/(n_s - 1)$ and the probability to remain in the same state is $T_{i,i} = 1 - \sum_{\forall j} T_{i,j}$. The normalization factor $(n_s - 1)$ guarantees that the probabilities sum to 1.

Example 4. Stochastic population dynamics: population with three strategies

The example below uses the same payoff structure of Example 2, i.e., we are looking at the 3 strategy Hawk-Dove-Human game. The result of this code is shown in Figure 4B. The only change with respect to Example 2 is that the function `plot_pairwise_comparison_rule_dynamics_in_simplex_without_roots` now requires the size of the population (Z) and the intensity of selection (β) as arguments, and will return an instance of the `PairwiseComparison` class.

```
import numpy as np
import matplotlib.pyplot as plt
from egttools.plotting.simplified import
    plot_pairwise_comparison_rule_dynamics_in_simplex_without_roots
from egttools.utils import calculate_stationary_distribution
# Parameters
payoffs = np.array([[-0.5, 2, 0], [0, 0, 2], [0, 1, 0]])
nb_strategies = 2; Z = 100; beta = 1; mu = 1/Z
type_labels = ['Hawk', 'Dove', 'Human']
# Create figure
fig, ax = plt.subplots(figsize=(10,8))
simplex, gradient_function, game, evolver =
    plot_pairwise_comparison_rule_dynamics_in_simplex_without_roots(payoff_matrix =
       payoffs, population_size = Z, beta = beta, ax = ax)
# Calculate stationary distribution
transitions = evolver.calculate_transition_matrix(beta = beta, mu = mu)
sd = calculate_stationary_distribution(transitions.transpose())
plot = (simplex
    .draw_triangle()
    .add_vertex_labels(type_labels, epsilon_bottom = 0.1, epsilon_top = 0.03)
    .draw_stationary_distribution(sd, alpha = 1, shrink = 0.5, edgecolors = 'gray', cmap =
       'binary', shading = 'gouraud', zorder = 0)
    .draw_gradients(zorder = 2, linewidth = 1.5)
    .add_colorbar(shrink = 0.5)
)
ax.axis('off')
ax.set_aspect('equal')
plt.xlim((-0.05,1.05))
plt.ylim((-0.02, simplex.top_corner +0.05))
```

Equipped with these tools, we can now compute the prevalence of each strategy through the stationary distribution p_e of the n_s-states Markov chain as described in the previous section. In this case, the stationary distribution characterizes the average time the population spends in each monomorphic state e. The stationary distribution can be used to estimate the average success (cooperation) level in the population. This can be done by multiplying the stationary distribution p_e by the probability of success (or cooperation) of each available strategy, i.e., the success $\eta = \boldsymbol{p_e H_e}$, where p_e is a row vector containing the stationary distribution, and H is a column vector containing the probability of success of each monomorphic state.

Finally, the fixation probabilities can also be used to draw an invasion diagram representing the transitions between the different monomorphic states (see Example 5 and Figure 6A). In such a diagram, arrows represent transitions favored by natural selection, i.e., those whose fixation probability exceeds $1/Z$ (associated with the fixation probability of a mutant under neutral evolution). This diagram provides a convenient measure of strategic stability, such that a strategy that has no outgoing arrows, i.e., which cannot be invaded by a mutant, is said to have a selective advantage. Such strategy is considered to be an Evolutionary Robust Strategy (ERS).[61]

Large-scale agent-based simulations

As explained in the previous section, analyzing a system with a large number of states can be intractable analytically. In those cases, we have to either resort to approximations to reduce the dimensionality of the system, or to numerical simulations. The latter are often based on Monte-Carlo methods used to estimate the relevant indicators of the population dynamics, e.g., the stationary distribution, the distribution of

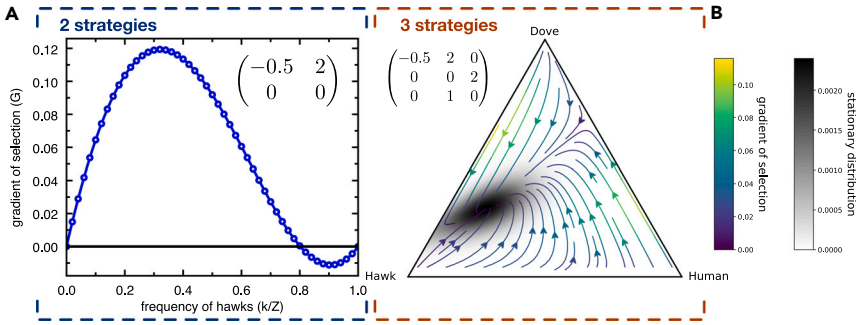


Figure 4. Gradient of selection of the stochastic birth-death process with pair-wise comparison for a 2 and 3 strategy Hawk-Dove game

Panel (A) shows the gradient of selection for a 2 strategy Hawk-Dove game. This plot information can be easily generated with the *plot_gradients* function in EGTtools (see Example 3). Panel (B) shows similar information for the three-strategy version of the Hawk-Dove game. In this case, the gradients are plot over a simplex S_3 (a triangle) and their intensity is shown using colors. In addition, we also plot in gray-scale the stationary distribution of the Markov chain that defines the dynamics of the system (see Example 4). ($Z = 100$, $\beta = 1$).

strategies, the wealth distribution, etc. Yet, these methods are computationally intensive and can require a long time to converge depending on the size of the system. For this reason, it is very important to use efficient implementations of such algorithms.

EGTtools provides efficient agent-based simulations through the *PairwiseComparisonNumerical* class implemented in C++, which is made accessible through the Python API. Figure 5A shows the error of estimating the stationary distribution of the Hawk-Dove game introduced in Example 3 using the method *estimate_stationary_distribution* of the *PairwiseComparisonNumerical* class, in comparison to the analytical calculations. In this figure, we display the expected frequency of doves, obtaining by multiplying the estimated stationary distribution by the fraction of doves present in each possible population state. It can be seen that the error is almost negligible for higher intensity of selection (β). It is normal that lower values of β issue more error because the system has more noise. This simply means that simulations with a higher number of generations (so that the system has time to converge to stationary states) are required. On average, the mean squared error is very small ($\approx 8 \times 10^{-6}$). Figure 5B shows the estimated stationary distribution for $\beta = 1$ in comparison to the distribution calculated analytically, and shows that the two are almost identical. Of course, this is a minimal example, with a population that can only adopt 2 strategies. In Example 6, and Figures 6B and 6C, we provide a more complex situation with 6 strategies

Other libraries

EGTtools is aimed at providing an integrated software framework to study the dynamics of multi-agent system through EGT as well as simplify the reproduction of EGT research. It is the only library available for Python, to the best of the authors' knowledge, which provides efficient C++ implementations of the numerical simulations which are required for analyzing large scale problems with EGT. It also provides an integrated set of visualization function and classes which are very useful for studying the dynamics of 2 and 3 strategy games. Nevertheless, there are other open-source libraries available which provide complementary functionalities.

In particular, the NashPy[62] and Axelrod[63] libraries are worth mentioning. NashPy provides several methods and functions to find and calculate game equilibria. Recently they have added methods to calculate evolutionary stable strategies and replicator dynamics. Axelrod is focused on the Iterated Prisoner's Dilemma (IPD) and aims providing an environment that facilitates the reproduction of research in this area. This library also includes methods to study population dynamics through the Moran process or the replicator equation. Yet, in regards to EGT, EGTtools provides a more complete set of tools that allows researchers to analyze and visualize the population dynamics of any Game and not only identify the equilibrium solutions.

Example 5. Invasion diagrams in the small mutation limit

As explained in the section about *dimensionality reduction*, if we want to study the dynamics of a population with a high number of possible strategies, we may end up with a large number of states, which can make it impractical (and depending on computing resources also impossible) to analyze analytically. In these cases, we have to either resort to large-scale computer simulations or to numerical approximations that facilitate the reduction of the number of states. One of these approximations is the limit of small mutations (when random changes in strategy of individuals in the population are rare).

The EGTtools analytical module implements this approximation in the `PairwiseComparison` class, through the `calculate_transition_and_fixation_matrix_sml` method, which returns two matrices: one containing a transition matrix, which returns the transition probabilities of the reduced Markov Chain assuming this small mutation limit (SML); and another containing the fixation probabilities of each column strategy *i* in a population of row strategy *j*. The stationary distribution can then be calculated using the `calculate_stationary_distribution` function implemented in `egttools.utils` with this transition matrix as input.

Figure 6A shows the result of the code below. The label inside each node represents the strategy adopted by all individuals of a monomorphic population. The number above/below the nodes represents the stationary distribution. An outgoing edge from a strategy A to B indicates that A is invaded by B. A dashed edge indicates that there is random drift ($\rho_N = 1/Z$) between the two strategies. The number between the edges indicates how many times the fixation probability is greater than random drift, N ρ.

```
from egttools.behaviors.Normal-Form.TwoActions import (Cooperator,Defector, TFT, Pavlov, GRIM, ↪ Random)
# Define which strategies can be present in the population
strategies = [Cooperator(), Defector(), TFT(), Pavlov(), Random(), GRIM()]
# define the game
payoffs = np.array([[1, 3],[0, 2]])
game = egt.games.NormalFormGame(100, payoffs, strategies)
# Define the parameters of the population and instantiate the evolver
evolver = egt.analytical.PairwiseComparison(Z, game)
# Calculate the transition and fixation probabilities and the stationary distribution
transition_matrix,fixation_probabilities = evolver.calculate_transition_and_fixation_matrix_sml(beta)
stationary_distribution = egt.utils.calculate_stationary_distribution(transition_matrix)
# Plot the invasion diagram
G = egt.plotting.draw_invasion_diagram([strategy.type().replace(''NFGStrategies:'', '') for strategy in
strategies], 1/Z, fixation_probabilities, stationary_distribution)
```

The only other Python library which is aimed at implementing EGT methods is DyPy (https://github.com/anjalika-nande/dynamics_sim).[64] This library implements the three most used types of processes used to study evolutionary game dynamics: Moran, Wright-Fisher and Replicator. It also implements

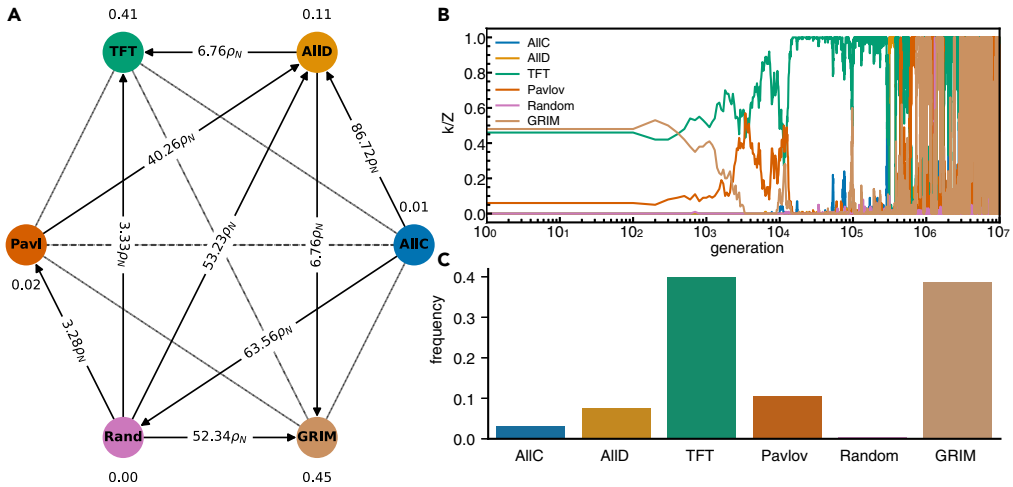


Figure 5. Estimation error of the stationary distribution in a Hawk-Dove game

In panel (A) we can see that the expected frequency of Doves estimated through numerical simulations is very close to the analytical calculations ($MSE = 9.965 \times 10^{-6}$). In panel (B) we plot the estimated stationary distribution for $\beta = 1$ and compare it to the one calculated analytically. The expected frequency of Doves shown in panel (A) can be calculated by multiplying the probability that the population will be in a given state (stationary distribution) by the frequency of Doves at that given state ($Z = 100$, $\mu = 10^{-3}$, all simulation have been run for 10^8 generations and the results are an average over 30 runs).

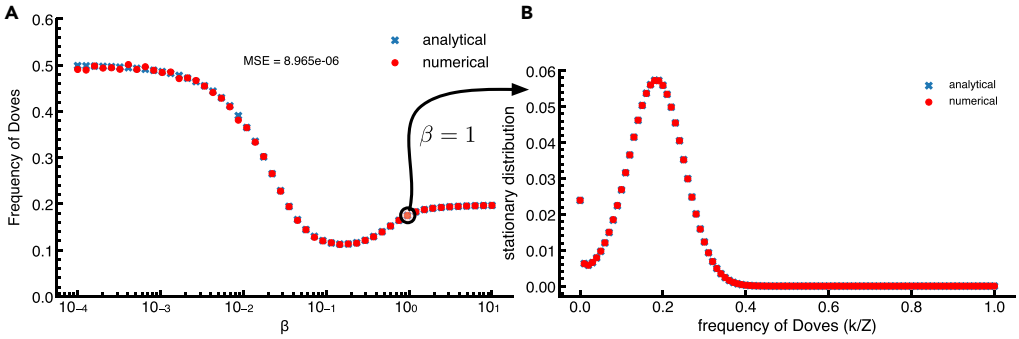


Figure 6. Finite population dynamics in the iterated prisoner's dilemma (IPD) with 6 strategies

Panel (A) shows the invasion diagram when considering the limit of small mutations (SML, see Example 5). Panel (B) shows the outcome of a single run of a numerical simulation of the Moran process with pair-wise imitation rule (see Example 6 line 9). The y axis shows the frequency of each strategy in the population (k/Z) at the generation shown in the x axis. Finally Panel (C) shows the distribution of strategies estimated through numerical simulations (see Example 6 line 12). We can observe that the estimated strategy distribution (considering a mutation rate $\mu = 10^{-3}$), panels (B) and (C)) is similar to the calculated assuming the SML. This can be an indicator that the SML assumption is acceptable for this problem ($Z = 100$, $\beta = 1$, all simulation have been run for 10^7 generations and the results are an average over 30 runs).

both individual and group-level selection and frequency biased imitation. However, it only supports numerical simulations of matrix-form games and is purely implemented in Python. In regards to this, we believe EGTtools provides a more complete functionality, allowing the analysis of any Game, with methods implemented more efficiently in C++, as well as the integration of both analytical and numerical methods.

Moreover, there are two other libraries aimed at the visualization of population dynamics with 3 strategies: egtsimplex (https://github.com/marvinboe/egtsimplex) and egtplot (https://github.com/mirzaevinom/egtplot). In particular, the egtsimplex inspired the implementation of our *Simplex2D* class.

Finally, we would like to mention the well-known Dynamo suite of Mathematica notebooks[65,66] with implementation of EGT methods, which is a great option for those researchers with access to a Mathematica license. We also refer to the excellent article of Hindersin et al.,[23] which provides an overview of computational methods in EGT and which was immensely helpful in the implementation of EGTtools.

Future implementations

EGTtools is a library in active development, and we plan to continue to add existing and new EGT models and methods to it. The current version of EGTtools is 0.1.12. We plan to add support for evolutionary

Example 6. Monte Carlo simulations

In this example, we consider the same Iterated Prisoner's Dilemma with 6 strategies used in the Example 5. We first show how to simulate a single run of the evolutionary process, and then, how to estimate the strategy distribution. We do not estimate the stationary distribution, because the number of possible population states in this example is too high and there are more than 3 dimensions. The results are plotted in Figures 6B and 6C. The independent runs of the simulations will be run in parallel in multiple processors (depending on the availability of the machine in which the simulation is being executed). We plot the results produced here in panels (b) and (c) of Figure 6.

```
# Simulation parameters Z = 100; nb_runs = 30; nb_generations = int(1e7); transitory =
int(1e3); beta = 1; mutation = ↪ 1e-3
# define the evolver
evolver = egt.numerical.PairwiseComparisonNumerical(Z, game, cache_size = 100000)
# Run a single simulation from a certain starting state
initial_state = [0, 0, 40, 10, 0, 50]
output = evolver.run(nb_generations, beta, mutation, initial_state)
# Estimate the distribution of strategies distribution = evolver.estimate_strategy_
distribution(nb_runs, nb_generations, transitory, beta, ↪ mutation)
```

simulations in complex networks in version 0.1.13, for multi-population models in version 0.1.14 and for group and multi-level selection in version 0.1.15. We will also add soon support for other competitive grown models such as the Moran process (with frequency based selection) and the Wright–Fisher process.

Limitations of the study

The version of EGTtools associated with this paper only implements the replicator equation as an evolutionary model for infinite populations, and the Moran process with pair-wise comparison for finite populations. In both cases, EGTtools allows the user to apply these models to both symmetric and asymmetric games (in the latter case it is assumed that all individuals in the population can adopt any strategy available), 2- and N-player games, and any number of strategies. Of course, as the number of strategies increase, analytical solutions become computationally complex to calculate. For this reason, EGTtools also offers the possibility of running numerical simulations efficiently (the numerical module is implemented in C++ and has been pre-compiled). However, EGTtools currently does not support EGT methods in structured populations (e.g., complex networks), nor other competitive growth models. These are important models in the EGT literature and we plan to add them progressively to the library, as noted in the section future implementations.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Structure of the library
  - How to implement a game in EGTtools
  - Installation of EGTtools
- QUANTIFICATION AND STATISTICAL ANALYSIS

ACKNOWLEDGMENTS

E.F.D. built this framework while being supported by an F.W.O. (Fonds Wetenschappelijk Onderzoek) Strategic Basic Research (SB) PhD grant (nr. G.1S639.17N), an F.R.S.-FNRS (Fonds de la Recherche Scientifique) grant (nr. 31257234), and an F.R.S-FNRS Chargé deRecherche grant (nr. 40005955). T.L. is supported by an FWO project (grant nr. G054919N) and two F.R.S.-FNRS PDR (grant numbers 31257234 and 40007793). E.F.D. and T.L. are supported by Service Public de Wallonie Recherche under grant nr. 2010235–ariac by digitalwallonia4.ai. T.L. acknowledges the support by the Flemish Government through the AI Research Program. F.C.S. acknowledges support by FCT-Portugal (grantsUIDB/50021/2020, PTDC/CCI-INF/7366/2020 and PTDC/MAT-APL/6804/2020). T.L. and F.C.S. both acknowledge the support by TAILOR, a project funded by EUHorizon 2020 research and innovation program under GA No 952215. Special thanks to Yannick Jadoul, author of Parselmouth, and Eugenio Bargiacchi, author of AIToolBox for their great advice on the implementation of EGTtools.

AUTHOR CONTRIBUTION

Conceptualization: E.F.D., F.C.S., and T.L.; Methodology: E.F.D., F.C.S., and T.L.; Software: E.F.D.; Validation: E.F.D.; Formal Analysis: E.F.D.; Investigation: E.F.D.; Resources: E.F.D.; Data Curation: E.F.D.; Writing – Original Draft: E.F.D., F.C.S., and T.L.; Writing – Review and Editing: E.F.D., F.C.S., and T.L.; Visualization: E.F.D.; Supervision: F.C.S. and T.L.; Project Administration: E.F.D.; Funding Acquisition: E.F.D. and T.L.

DECLARATION OF INTERESTS

The authors declare no competing interests.

INCLUSION AND DIVERSITY

We support inclusive, diverse, and equitable conduct of research.

Received: November 22, 2022
Revised: February 17, 2023
Accepted: March 13, 2023
Published: March 17, 2023

REFERENCES

1. Osborne, M.J., and Rubinstein, A. (1994). A Course in Game Theory (MIT press).

2. Nash, J.F., Jr. (1950). Equilibrium points in n-person games. Proc. Natl. Acad. Sci. USA *36*, 48–49. https://doi.org/10.1073/pnas.36.1.48.

3. Sigmund, K. (2010). The Calculus of Selfishness (Princeton University Press). https://doi.org/10.1038/4641280a.

4. Nowak, M.A. (2006). Evolutionary Dynamics: Exploring the Equations of Life (Harvard university press).

5. Traulsen, A., Nowak, M.A., and Pacheco, J.M. (2006). Stochastic dynamics of invasion and fixation. Phys. Rev. E - Stat. Nonlinear Soft Matter Phys. *74*, 011909. https://doi.org/10.1103/PhysRevE.74.011909.

6. Van Segbroeck, S., De Jong, S., Nowé, A., Santos, F.C., Lenaerts, T., Jong, S.d., Nowé, A., Santos, F.C., and Lenaerts, T. (2010). Learning to coordinate in complex networks. Adapt. Behav. *18* (5), 416–427. https://doi.org/10.1177/1059712310384282.

7. Traulsen, A., and Nowak, M.A. (2007). Chromodynamics of cooperation in finite populations. PLoS One *2*, e270. https://doi.org/10.1371/journal.pone.0000270.

8. Han, T.A., and Lenaerts, T. (2016). A synergy of costly punishment and commitment in cooperation dilemmas. Adapt. Behav. *24* (4), 237–248. https://doi.org/10.1177/1059712316653451.

9. Oechssler, J., and Riedel, F. (2001). Evolutionary dynamics on infinite strategy spaces. Econ. Theor. *17*, 141–162. https://doi.org/10.1007/PL00004092.

10. Sigmund, K. (2012). Moral assessment in indirect reciprocity. J. Theor. Biol. *299*, 25–30. https://doi.org/10.1016/j.jtbi.2011.03.024.

11. Nowak, M.A., and Sigmund, K. (1998). The dynamics of indirect reciprocity. J. Theor. Biol. *194*, 561–574. https://doi.org/10.1006/jtbi.1998.0775.

12. Sasaki, T., Okada, I., and Nakai, Y. (2017). The evolution of conditional moral assessment in indirect reciprocity. Sci. Rep. *7*, 41870–41878. https://doi.org/10.1038/srep41870.

13. Santos, F.C., Pacheco, J.M., and Skyrms, B. (2011). Co-evolution of pre-play signaling and cooperation. J. Theor. Biol. *274*, 30–35. https://doi.org/10.1016/j.jtbi.2011.01.004.

14. Martinez-Vaquero, L.A., Santos, F.C., and Trianni, V. (2020). Signalling boosts the evolution of cooperation in repeated group interactions. J. R. Soc. Interface *17*, 20200635. https://doi.org/10.1098/rsif.2020.0635.

15. Santos, F.P., Santos, F.C., and Pacheco, J.M. (2016). Social norms of cooperation in small-scale societies. PLoS Comput. Biol. *12*, e1004709. https://doi.org/10.1371/journal.pcbi.1004709.

16. Zisis, I., Di Guida, S., Han, T.A., Kirchsteiger, G., and Lenaerts, T. (2015). Generosity motivated by acceptance–evolutionary analysis of an anticipation game. Sci. Rep. *5*, 18076. https://doi.org/10.1038/srep18076.

17. Szabó, G., and Fáth, G. (2007). Evolutionary games on graphs. Phys. Rep. *446*, 97–216. https://doi.org/10.1016/j.physrep.2007.04.004.

18. Scatà, M., Di Stefano, A., La Corte, A., Liò, P., Catania, E., Guardo, E., and Pagano, S. (2016). Combining evolutionary game theory and network theory to analyze human cooperation patterns, Chaos. Chaos, Solit. Fractals *91*, 17–24. https://doi.org/10.1016/j.chaos.2016.04.018.

19. Santos, F.C., Pacheco, J.M., and Lenaerts, T. (2006). Cooperation prevails when individuals adjust their social ties. PLoS Comput. Biol. *2*, e140. https://doi.org/10.1371/journal.pcbi.0020140.

20. Van Segbroeck, S., Santos, F.C., Nowé, A., Pacheco, J.M., and Lenaerts, T. (2008). The evolution of prompt reaction to adverse ties. BMC Evol. Biol. *8*, 287–288. https://doi.org/10.1186/1471-2148-8-287.

21. Van Segbroeck, S., Santos, F.C., Pacheco, J.M., and Lenaerts, T. (2010). Coevolution of cooperation, response to adverse social ties and network structure. Games *1*, 317–337. https://doi.org/10.3390/g1030317.

22. Wang, Z., Szolnoki, A., and Perc, M. (2013). Interdependent network reciprocity in evolutionary games. Sci. Rep. *3*, 1183–1187. https://doi.org/10.1038/srep01183.

23. Hindersin, L., Wu, B., Traulsen, A., and García, J. (2019). Computation and simulation of evolutionary game dynamics in finite populations. Sci. Rep. *9*, 6946. https://doi.org/10.1038/s41598-019-43102-z.

24. E. Fernández Domingos (2022). EGTtools: Toolbox for evolutionary game theory. Preprint at Zenodo. https://doi.org/10.5281/zenodo.3687125.

25. Santos, F.C., and Pacheco, J.M. (2011). Risk of collective failure provides an escape from the tragedy of the commons. Proc. Natl. Acad. Sci. USA *108*, 10421–10425. https://doi.org/10.1073/pnas.1015648108.

26. Smith, J.M. (1974). The theory of games and the evolution of animal conflicts. J. Theor. Biol. *47*, 209–221. https://doi.org/10.1016/0022-5193(74)90110-6.

27. Smith, J.M. (1993). The Theory of Evolution (Cambridge University Press).

28. Hill, K. (2002). Altruistic cooperation during foraging by the ache, and the evolved human predisposition to cooperate. Hum. Nat. *13*, 105–128. https://doi.org/10.1007/s12110-002-1016-3.

29. Kaplan, H., Hill, K., Lancaster, J., and Hurtado, A.M. (2000). A theory of human life history evolution: diet, intelligence, and longevity. Evol. Anthropol.: Issues, News, and Reviews: Issues, News, and Reviews *9*, 156–185. https://doi.org/10.1002/1520-6505(2000)9:4¸156::AID-EVAN5¿3.0.CO;2-7.

30. Ostrom, E. (1990). Governing the Commons: The Evolution of Institutions for Collective Action (Cambridge university press).

31. Wang, Z., Jusup, M., Guo, H., Shi, L., Geček, S., Anand, M., Perc, M., Bauch, C.T., Kurths, J., Boccaletti, S., and Schellnhuber, H.J. (2020). Communicating sentiment and outlook reverses inaction against collective risks. Proc. Natl. Acad. Sci. USA *117*, 17650–17655. https://doi.org/10.1073/pnas.1922345117.

32. Chen, X., Szolnoki, A., and Perc, M. (2012). Risk-driven migration and the collective-risk social dilemma. Phys. Rev. E - Stat. Nonlinear Soft Matter Phys. *86*, 036101. https://doi.org/10.1103/PhysRevE.86.036101.

33. Hilbe, C., Abou Chakra, M., Altrock, P.M., and Traulsen, A. (2013). The evolution of strategic timing in collective-risk dilemmas. PLoS One *8*, e66490. https://doi.org/10.1371/journal.pone.0066490.

34. Santos, F.C., Vasconcelos, V.V., Santos, M.D., Neves, P.N.B., and Pacheco, J.M. (2012). Evolutionary dynamics of climate change under collective-risk dilemmas. Math. Model Methods Appl. Sci. *22* (supp01), 1140004. https://doi.org/10.1142/S0218202511400045.

35. Vasconcelos, V.V., Santos, F.C., and Pacheco, J.M. (2013). A bottom-up institutional approach to cooperative governance of risky commons. Nat. Clim. Change *3*, 797–801. https://doi.org/10.1038/nclimate1927.

36. Vasconcelos, V.V., Santos, F.C., and Pacheco, J.M. (2015). Cooperation dynamics of polycentric climate governance. Math. Model Methods Appl. Sci. *25*, 2503–2517. https://doi.org/10.1142/S0218202515400163.

37. Pacheco, J.M., Santos, F.C., Souza, M.O., and Skyrms, B. (2009). Evolutionary dynamics of collective action in n-person stag hunt dilemmas. Proc. Biol. Sci. *276*, 315–321. https://doi.org/10.1098/rspb.2008.1126.

38. Fudenberg, D., and Imhof, L.A. (2006). Imitation processes with small mutations. J. Econ. Theor. *131*, 251–262. https://doi.org/10.1016/j.jet.2005.04.006.

39. Vasconcelos, V.V., Santos, F.P., Santos, F.C., and Pacheco, J.M. (2017). Stochastic dynamics through hierarchically embedded Markov chains. Phys. Rev. Lett. *118*, 058301. https://doi.org/10.1103/PhysRevLett.118.058301.

40. Imhof, L.A., Fudenberg, D., and Nowak, M.A. (2005). Evolutionary cycles of cooperation and defection. Proc. Natl. Acad. Sci. USA *102*, 10797–10800. https://doi.org/10.1073/pnas.0502589102.

41. Karlin, S., and Taylor, H.M.A. (1975). A First Course in Stochastic Processes, 2nd Edition (New York: Academic Press).

42. Santos, F.C., Santos, M.D., and Pacheco, J.M. (2008). Social diversity promotes the emergence of cooperation in public goods games. Nature *454*, 213–216. https://doi.org/10.1038/nature06940.

43. Wang, J., Wu, B., W C Ho, D., and Wang, L. (2011). Evolution of cooperation in multilevel public goods games with community structures. EPL *93*, 58001. https://doi.org/10.1209/0295-5075/93/58001.

44. Pinheiro, F.L., Vasconcelos, V.V., Santos, F.C., and Pacheco, J.M. (2014). Evolution of all-or-none strategies in repeated public goods dilemmas. PLoS Comput. Biol. *10*, e1003945. https://doi.org/10.1371/journal.pcbi.1003945.

45. Martinez-Vaquero, L.a., Han, T.A., Pereira, L.M., and Lenaerts, T. (2015). Apology and forgiveness evolve to resolve failures in cooperative agreements. Sci. Rep. *5*, 10639. https://doi.org/10.1038/srep10639.

46. Hauser, O.P., Hendriks, A., Rand, D.G., and Nowak, M.A. (2016). Think global, act local: preserving the global commons. Nature *2016*, 1–44. https://doi.org/10.1038/srep36079.

47. Wang, J., Fu, F., Wu, T., and Wang, L. (2009). Emergence of social cooperation in threshold public goods games with collective risk. Phys. Rev. E - Stat. Nonlinear Soft Matter Phys. *80*, 016101–016111. https://doi.org/10.1103/PhysRevE.80.016101.

48. Souza, M.O., Pacheco, J.M., and Santos, F.C. (2009). Evolution of cooperation under n-person snowdrift games. J. Theor. Biol. *260*, 581–588. https://doi.org/10.1016/j.jtbi.2009.07.010.

49. Greenwood, G. (2011). Evolution of strategies for the collective-risk social dilemma relating to climate change. EPL *95*, 40006. https://doi.org/10.1209/0295-5075/95/40006.

50. Pacheco, J.M., Vasconcelos, V.V., and Santos, F.C. (2014). Climate change governance, cooperation and self-organization. Phys. Life Rev. *11*, 573–586. https://doi.org/10.1016/j.plrev.2014.02.003.

51. Vasconcelos, V.V., Santos, F.C., Pacheco, J.M., and Levin, S.A. (2014). Climate policies under wealth inequality. Proc. Natl. Acad. Sci. USA *111*, 2212–2216. https://doi.org/10.1073/pnas.1323479111.

52. Santos, F.P., Mascarenhas, S., Santos, F.C., Correia, F., Gomes, S., and Paiva, A. (2020). Picky losers and carefree winners prevail in collective risk dilemmas with partner selection. Auton. Agent. Multi. Agent. Syst. *34*, 40. https://doi.org/10.1007/s10458-020-09463-w.

53. Han, T.A., Pereira, L.M., and Santos, F.C. (2011). The role of intention recognition in the evolution of cooperative behavior. In IJCAI International Joint Conference on Artificial Intelligence, pp. 1684–1689. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-283.

54. Han, T., Tran-Thanh, L., and Jennings, N. (2014). The cost of interference in evolving systems. In Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), pp. 1719–1720.

55. Santos, F.P., Pacheco, J.M., Paiva, A., and Santos, F.C. (2019). Evolution of collective fairness in hybrid populations of humans and agents. Proc. AAAI Conf. Artif. Intell. *33*, 6146–6153. https://doi.org/10.1609/aaai.v33i01.33016146.

56. Grujić, J., Fosco, C., Araujo, L., Cuesta, J.A., and Sánchez, A. (2010). Social experiments in the mesoscale: humans playing a spatial prisoner's dilemma. PLoS One *5*, e13749. https://doi.org/10.1371/journal.pone.0013749.

57. Grujić, J., Cuesta, J.A., and Sánchez, A. (2012). On the coexistence of cooperators, defectors and conditional cooperators in the multiplayer iterated Prisoner's Dilemma. J. Theor. Biol. *300*, 299–308. https://doi.org/10.1016/j.jtbi.2012.02.003.

58. Santos, F.P., Pacheco, J.M., Paiva, A., and Santos, F.C. (2019). Evolution of collective fairness in hybrid populations of humans and agents. Proc. AAAI Conf. Artif. Intell. *33*, 6146–6153.

59. Domingos, E.F., Grujić, J., Burguillo, J.C., Kirchsteiger, G., Santos, F.C., and Lenaerts, T. (2020). Timing uncertainty in collective risk dilemmas encourages group reciprocation and polarization. iScience *23*, 101752. https://doi.org/10.1016/j.isci.2020.101752.

60. Ewens, W.J. (2012). Mathematical Population Genetics, 2nd Edition (Springer Science+Business Media, LLC). https://doi.org/10.1007/978-0-387-21822-9.

61. Stewart, A.J., and Plotkin, J.B. (2013). From extortion to generosity, evolution in the Iterated Prisoner's Dilemma. Proc. Natl. Acad. Sci. USA *110*, 15348–15353. https://doi.org/10.1073/pnas.1306246110.

62. Knight, V., and Campbell, J. (2018). Nashpy: a python library for the computation of nash equilibria. J. Open Source Softw. *3*, 904. https://doi.org/10.21105/joss.00904.

63. Knight, V., Campbell, O., Marc, Gaffney, T., Shaw, E., Janga, V.R., Glynatsi, N., Campbell, J., Langner, K.M., Singh, S., et al. (2021). Axelrod-python/axelrod: v4.12.0. Preprint at Zendo. https://doi.org/10.5281/zenodo.5616793.

64. Nande, A., Ferdowsian, A., Lubin, E., Yoeli, E., and Nowak, M. (2020). Dypy: A python library for simulating matrix-form games. Preprint at arXiv. https://doi.org/10.48550/arXiv.2007.13815.

65. Sandholm, W.H., Dokumaci, E., and Franchetti, F. (2012). Dynamo: Diagrams for Evolutionary Game Dynamics.

66. Franchetti, F., and Sandholm, W.H. (2013). An introduction to dynamo: diagrams for evolutionary game dynamics. Biol. Theory *8*, 167–178. https://doi.org/10.1007/s13752-013-0109-z.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
EGTtools v0.1.12	https://github.com/Socrats/EGTTools	https://doi.org/10.5281/zenodo.7458631
EGTtools code examples	This paper	https://github.com/Socrats/egt-tutorial
NumPy v1.24.1	https://github.com/numpy/numpy	https://github.com/numpy/numpy
Matplotlib v3.5.2	https://matplotlib.org	https://matplotlib.org
SciPy v1.8.1	https://scipy.org	https://scipy.org
NetworkX v2.6.3	https://networkx.org	https://networkx.org
Seaborn v0.11.2	https://seaborn.pydata.org	https://seaborn.pydata.org
Other		
EGTtools Documentation	https://egttools.readthedocs.io	https://egttools.readthedocs.io

RESOURCE AVAILABILITY

Lead contact

Further information and any related requests should be directed to and will be fulfilled by the lead contact, Elias Fernández Domingos (elias.fernandez.domingos@ulb.be).

Materials availability

This study did not generate new unique reagents.

Data and code availability

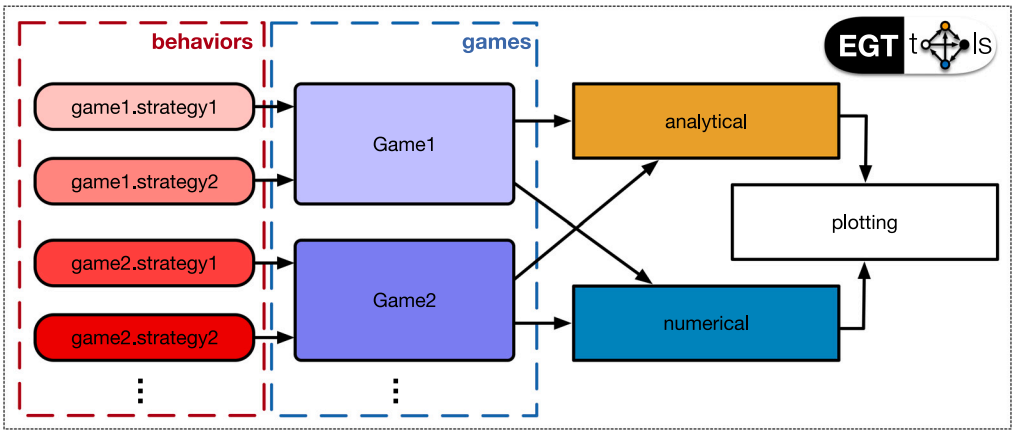
- All data used to produce the figures in this paper can be reproduced using the EGTtools code examples deposited in the *egt-tutorial* repository listed in the key resources table.
- All original code has been deposited at Github and Zenodo and is publicly available. DOIs are listed in the key resources table.

METHOD DETAILS

In the following sections we add some extra information that will be helpful in using EGTtools. You may also find the detailed documentation of the library at https://egttools.readthedocs.io.

Structure of the library

EGTtools implements several analytical and numerical evolutionary game theoretical models and methods (see Figure). These are contained in the *analytical* and *numerical* modules. The implementations in these models require a *Game* object which will provide the payoff of each strategy for each possible game configuration. In pairwise games, this will be a square matrix giving the payoff of each possible strategy against each other, while in n-player games, this matrix should return the payoff of a strategy for each possible group composition (see the following section for more information on how games are implemented). Finally, the strategies are implemented in the *behaviors* module and should be assigned to a *Game*, since the rules of the game, and the action space, will limit which strategies can be used. Finally, EGTtools provides a *plotting* module to visualize the results of the analytical and numerical models.



Schema of EGTtools

The package is structured into five main modules: a games module which defines the environment of the strategic interactions, a behaviors module, which define the strategies that can be used in each game, an analytical and a numerical modules which implement evolutionary game theoretical methods and models, and, finally, a plotting module which contains several functions and classes which can be used to visualize the results of your models.

How to implement a game in EGTtools

The *Game* class is core to the EGTtools library, as it defines the environment in which strategic interactions take place. All *games* must extend the *AbstractGame* abstract class. This class nine methods: play, calculate_payoffs, calculate_fitness, __str__, nb_strategies, type, payoffs, payoff, save_payoffs. Below you can see a description of this class in Python and the purpose of each method:

```
class AbstractGame:
    def play(self, group_composition: Union[List[int], numpy.ndarray], game_
    payoffs: numpy.ndarray) -> None:
        """
        Calculates the payoff of each strategy inside the group.
        Parameters
        ————————
        group_composition: Union[List[int], numpy.ndarray]
        counts of each strategy inside the group.
        game_payoffs: numpy.ndarray
        container for the payoffs of each strategy
        """
        pass
    def calculate_payoffs(self) -> np.ndarray:
    """
    This method should set a numpy.ndarray called self.payoffs_with the expected
    payoff of each strategy in each possible state of the game
    """
        pass
    def calculate_fitness(self, strategy_index: int, pop_size: int, population_
    state: numpy.ndarray) -> float:
```

```
    """
    This method should return the fitness of strategy
    with index `strategy_index` for the given `population_state`.
    """
        pass
    def __str__(self) -> str:
    """
    This method should return a string representation of the game.
    """
        pass
    def nb_strategies(self) -> int:
    """
    This method should return the number of strategies which can play the game.
    """
        pass
    def type(self) -> str:
    """
    This method should return a string representing the type of game.
    """
        pass
    def payoffs(self) -> np.ndarray:
    """
    This method should return the payoff matrix of the game, which gives the payoff of
    each strategy in each given context.
    """
        pass
    def payoff(self, strategy: int, group_configuration: List[int]) -> float:
    """
    This method should return the payoff of a strategy for a given `group_configura-
    tion`, which gives the counts of each strategy in the group. This method only
    needs to be implemented for N-player games
    """
        pass
    def save_payoffs(self, file_name: str) -> None:
    """
    This method should implement a mechanism to save the payoff matrix and parame-
    ters of the game to permanent storage.
    """
        pass
```

However, in most scenarios the fitness of a strategy at a given population state is its expected payoff at that state. For this reason, EGTtools provides two other abstract classes to simplify the implementation of new games: egttools.games.AbstractTwoPLayerGame, for two-player games; and

egttools.games.AbstractNPlayerGame for N-player games. When using these abstract classes, you only need to implement two methods: play and calculate_payoffs. Below you can find an example on how to implement the N-player Stag Hunt game from[37] with this latter abstract class:

```
class NPlayerStagHunt(AbstractNPlayerGame):
    def __init__(self, group_size, enhancement_factor, cooperation_threshold,
    cost):
        super().__init__(2, group_size) # Must be initialized at the beginning!
        self.group_size_= group_size # N
        self.enhancement_factor_= enhancement_factor # F
        self.cooperation_threshold_= cooperation_threshold # M
        self.cost_= cost # c
        self.strategies = ['Defect', 'Cooperate']
        self.nb_strategies_= 2
        self.nb_group_configurations_= self.nb_group_configurations()
        self.calculate_payoffs()
    def play(self, group_composition: Union[List[int], np.ndarray], game_payoffs:
    np.ndarray) -> None:
        if group_composition[0] == 0:
            game_payoffs[0] = 0
            game_payoffs[1] = self.cost_* (self.enhancement_factor_- 1)
        elif group_composition[1] == 0:
            game_payoffs[0] = 0
            game_payoffs[1] = 0
        else:
            game_payoffs[0] = ((group_composition[1] * self.enhancement_factor_)
            self.group_size_) if group_composition[
            1]->= self.cooperation_threshold_else 0 # ↪ Defectors
            game_payoffs[1] = game_payoffs[0] - self.cost_# Cooperators
    def calculate_payoffs(self) -> np.ndarray:
        payoffs_container = np.zeros(shape=(self.nb_strategies_,), dtype=np.
        float64)
        for i in range(self.nb_group_configurations_):
            # Get group composition
            group_composition = sample_simplex(i, self.group_size_, self.nb_
            strategies_)
            self.play(group_composition, payoffs_container)
            for strategy_index, strategy_payoff in enumerate(payoffs_container):
            self.update_payoff(strategy_index, i, strategy_payoff)
            # Reinitialize payoff vector
            payoffs_container[:] = 0
        return self.payoffs()
```

Installation of EGTtools

EGTtools can be installed using PyPi on Linux, macOS, and Windows:

pip install egttools

To update your installed version to the latest release, add `-U` (or `–upgrade`) to the command:

pip install-U egttools

EGTtools can also be built from source. For that you need to follow the following steps:

To install all required packages run:

python-m venv egttools-env source egttools-env/bin/activate pip install-r requirements.txt

Or with anaconda:

conda env create-f environment.yml conda activate egttools-env

Then, you can build the package with:

pip install build cd <path> python-m build

Where <path> represents the path to the EGTtools folder. If you are running this while inside the EGTtools folder, then <path> is simply `./`. Finally, you can also install EGTtools with:

python-m pip install <path>

QUANTIFICATION AND STATISTICAL ANALYSIS

In Figure 5A, we calculate the mean squared root error between the analytical calculation of the expected frequency of Doves, Y_i, and the numerical estimation, $\widehat{Y}_i$. This error is calculated by summing the squared differences between each of the sampled points (n=50) and averaging by the number points according to the following equation: $\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2$.