Social Learning Strategies Tournament II

Rules for entry

Luke Rendell and Kevin Laland, University of St Andrews

Introduction

In recent years there has been growing interest (spanning several research fields, but especially economics, anthropology and biology), in the problem of how best to acquire valuable information from others. Mathematical and computational solutions to this problem are starting to emerge, often using game-theoretical approaches. The first Social Learning Strategies Tournament, inspired by Robert Axelrod's famous Prisoner's Dilemma tournament on the evolution of cooperation (Axelrod 1980), was a great success, with over a hundred entries from researchers in 16 different academic disciplines, and 15 different countries, and with the findings published in *Science* (Rendell et al. 2010). The competition increased understanding of, and stimulated research on, social learning strategies, as Axelrod's tournament did for research on the evolution of cooperation. While it was natural to start with a comparatively simple competition, the rules of the first tournament prevented certain types of learning strategy (e.g. model-based biases) from being deployed, and did not consider how strategy use would be affected by spatial variation or cumulative culture. We have now received funding to organize a second tournament from the European Commission, which will relax these constraints and allow these issues to be investigated. Once again, we seek to solicit entries from a broad range of researchers, and to generate insights into the nature of human learning and culture that will lead to high-profile publications.

Background

It is commonly assumed that social learning is inherently worthwhile. It is widespread in nature (Hoppitt and Laland 2008). Individuals are thought to benefit by copying because they take a short cut to acquiring adaptive information, saving themselves the costs of asocial (e.g. trial-and-error) learning. Copying, it is assumed, has the advantage that individuals do not need to re-invent technology, devise novel solutions, or evaluate environments for themselves. Intuitive though this argument may be, it is flawed (Boyd and Richerson 1985, 1995; Giraldeau et al. 2002). It does not explain how copying *per se* might be a recipe for success. This is easy to understand if social learning is regarded as a form of parasitism on information (Giraldeau et al. 2002): asocial learners are information *producers*, while social learners are information *scroungers*. Game-theoretical models of producer-scrounger interactions reveal that scroungers do better than producers only when fellow scroungers are rare, while at equilibrium their payoffs are equal (Barnard and Sibly 1981). Similarly, theoretical analyses of the evolution of social learning in a changing environment (e.g. Boyd and Richerson 1985; Rogers 1988; Boyd and Richerson 1995; Feldman et al. 1996) reveal that social learners have higher fitness than asocial learners when copying is rare, because most 'demonstrators' are asocial learners who will have sampled accurate information about the environment at some cost. As the frequency of social learning increases, the value of copying declines, because the proportion of asocial learners producing reliable information appropriate to the observer is decreasing. Equilibrium is reached with a mixture of social and asocial learning (Enquist et al. 2007). These mathematical analyses, together with more conceptual theory (e.g. Galef Jr. 1995), imply that copying others indiscriminately is not adaptive; rather, individuals must use social learning *selectively*, and learn asocially some of the time. Natural selection in animals capable of social learning ought to have fashioned specific adaptive *social learning strategies* that dictate the circumstances under which individuals will exploit information provided by others (Boyd and Richerson 1985; Schlag 1998; Henrich and McElreath 2003; Laland 2004). At present, it is not clear which social learning strategy, if any, is best. This tournament is the second to have been set up to address this question.

The first social learning strategies tournament was a computer-based competition in which entrants submitted a strategy specifying the best way for agents living in a simulated environment to learn (Rendell et al. 2010; Rendell et al. 2011). The simulation environment was a multi-armed bandit with 100 possible behaviour patterns that an agent could learn and subsequently exploit. Each behaviour had a payoff, drawn from an exponential distribution, and the payoff could change over time. This simulated environment contained a population of 100 agents, each controlled by one of the strategies entered into the tournament.

In each model iteration, agents selected, using a decision process specified by their strategy, one of three moves. The first, INNOVATE, resulted in an agent learning the identity and payoff of one new behaviour, selected at random. The second, EXPLOIT, represented an agent choosing to perform a behaviour it already knew and receiving the payoff associated with that behaviour. The third, OBSERVE, represented an agent observing one or more of those agents who chose to play EXPLOIT, and learning the identity and payoff of the behaviour the observed agent was performing. The fitness of agents was determined by the total payoff received divided by the number of iterations through which they had lived. Evolution occurred through a death–birth process, with dying agents replaced by the offspring of survivors; the probability of reproduction was proportional to fitness.

The most important finding from the first tournament was the success of strategies that relied almost entirely on copying (i.e. OBSERVE) to learn behaviour. It revealed that copying pays under a far greater range of conditions than previously thought, even when extremely error-prone. In any given simulation involving the top-performing strategies, very little of the learning performed was asocial and learning for the winning strategy (DISCOUNTMACHINE, submitted by Dan Cownden and Tim Lillicrap; Figure 1) was almost exclusively social. The strength of this result depends in part on the tournament assumption that individuals build up a repertoire of multiple behavior patterns, rather than focussing on a single acquired behaviour, as in most analytical theory. This meant that when a copied behaviour turned out to confer low fitness, agents could switch rapidly to an alternative behaviour in the repertoire, thereby removing one of the drawbacks to copying identified in the analytical literature.

And so the prize of €10,000 goes to
Dan Cownden and Tim Lillicrap

Figure 1: Winners of the first Social Learning Strategies Tournament, Dan Cownden and Tim Lillicrap, receiving their cheque for €10,000. The winners were also co-authors of the tournament publication in *Science*.

The tournament also highlighted the role of copied individuals as filters of information. Previous theory had placed the onus on learners to perform this adaptive filtering, demanding selectivity, and therefore specific cognitive capabilities, on the part of the copier (e.g. Enquist et al. 2007). However, the tournament established that even non-selective copying is beneficial relative to asocial learning, because copied individuals typically perform the highest payoff behavior in their repertoire, thereby generating a non-random sample of highperformance behaviour for others to copy. These insights go some way to explaining the discrepancy between Rogers' (Rogers 1988) analysis and the empirical fact of human reliance on social information (known as 'Rogers' paradox'). They also help to explain why social learning is so widespread in nature, observed not just in primates and birds (Hoppitt and Laland 2008), but even in fruit flies and crickets (Leadbeater and Chittka 2007): even indiscriminate copying is generally more efficient than trial-and-error learning. However, because of its design, the first tournament provided no information on the issue of from whom one should learn, an issue that we address in the second tournament. We also consider how strategy use is affected by the opportunity for cumulative cultural learning and spatial variation in the environment.

Objective for entrants

The Second Social Learning Strategies Tournament extends the first in three ways, and entrants can choose to engage with any one, or all, of these extensions. In the first extension, individuals that choose to learn from others will be given a choice of from whom they want to learn, as well as information about who is available to learn from, in order to guide that choice. In the second, we will allow individuals to invest time in cumulatively improving a behaviour they already know (a move we call REFINE) in order to increase its payoff. Finally, the third extension is to introduce spatial variation in the environment by simulating a meta-population comprising three demes, as opposed to a single population. For brevity, we henceforth refer to these extensions as the *model-bias*, *cumulative*, and *spatial* extensions.

As with the first Social Learning Strategies Tournament, strategies will be tested in computational evolutionary simulations. The specification of the simulations, details on how to enter, and detailed tournament rules are given below. Entrants should ensure they are familiar with this material, as the details given are crucial in ensuring that your strategy will be considered in the tournament.

To enter the tournament, applicants need to specify a *strategy* detailing how individuals should acquire and use information in a simulated environment that varies in space and time. The strategy corresponds to a set of rules that specify when an individual should use a behaviour it already knows (EXPLOIT), when it should engage in trial-and-error learning (INNOVATE), when it should learn from other individuals (OBSERVE) and, in the case of the cumulative extension, when it should invest in improving a behaviour it already knows (REFINE). Performing the right behaviour is important, as fitness depends on how well behaviour is matched to the current environment. However, learning is not free, as there is a time cost incurred whenever an individual learns or refines behaviour.

Entries comprise the specification of up to two computational procedures that define a strategy. The first, termed *move*, takes in information about an individual's life so far, and returns a decision about whether to INNOVATE, OBSERVE, EXPLOIT or REFINE. The second, which only need be defined for entries that engage with the *model-bias* extension, is termed *observe_who*, and, in the event *move* decides on OBSERVE, takes in information about the individuals that are available to copy and decides which of them it wishes to copy.

Strategy evaluation

The competition will be run in three stages.

Stage 1: *Single extension pairwise*
All valid entered strategies will take part in three sets of round-robin contests between all pairs of entered strategies, with each set of contests enacting one, and only one, of the three extensions (i.e. a model-bias set, a cumulative-learning set, and a spatial-structure set). A contest, say between strategies A and B, involves exploring whether strategy A can invade a population containing only strategy B, and vice-versa. Each contest will involve replicate simulations, with each strategy as the invader 50% of the time. In each simulation, after a fixed number of iterations, the frequency of each strategy (that is, the proportion of the population with that strategy) will be recorded, and the average frequency across repetitions will be the score of that strategy in that contest.

Stage 2: *Single extension melee*
From each *single extension pairwise* contest set, (at least) the ten highest scoring strategies will be entered into a *melee* with the same extension enacted. Within each of the three *melees* all strategies compete simultaneously in multiple simulations across a broad range of conditions. In each simulation, after a fixed number of rounds, the frequency of each strategy will be the score for that strategy. The strategy with the highest average score in each *single extension melee* will be declared the winner of that single extension contest and win a prize. Participants should therefore try and construct their strategies so that they are likely to work well under most conditions. Thus at the end of stage 2 we will have identified the (up to) three strategies that operate most effectively in the specialized model-bias, cumulative and spatial contest circumstances.

Stage 3: *All extensions melee*
Finally, at least the five highest performing strategies from each *single extension melee* will go forward to compete in a further series of *melee* simulations in which all three extensions are simultaneously enacted. We may also include up to 5 ‘wildcard’ entries, in the form of generalist strategies that performed well across all three contests combined, but did not make it into the top 5 in any specialist contest (see 2.3 below). Multiple simulations will be run over a range of conditions, and the strategy with the highest average score in these simulations will be declared the overall tournament champion.

TECHNICAL DETAILS[1]

1. Simulation specifications

Each simulation will contain either a single population, or, in the *spatial* case, three populations (demes) of 100 individuals, and will run for up to 10,000 rounds[2]. A single round will consist of the following computational steps:

(i) Individuals are selected sequentially to choose a move (see **1.2** below) until all individuals have played.
(ii) Individuals reproduce with probabilities proportional to their average lifetime payoffs (see **1.3** below).
(iii) The environment may change, with a given probability (see **1.1** below).
(iv) In the *spatial* case, some individuals will migrate between populations (see **1.3** below)

1.1 Environment and behaviour

1.1.1 Each population has an associated environment, represented as a 'multi-arm bandit', wherein actors select from a range of possible behavioural acts and receive a payoff associated with that act. There will be 100 possible acts, and the payoff for each act will be chosen at the start of each simulation from a distribution with many relatively small payoffs and some relatively large ones[3]. Therefore the environment can be represented by a table with two rows associating behavioural acts with payoffs, for example:

Act:	1	2	3	4	5	…	100
Payoff:	4	0	17	1	7	…	3

1.1.2 Payoffs are not constant, and the payoff associated with a given behavioural act will change between each round of the simulation with a certain probability. This probability is a parameter of the simulation, called p_c (see **1.4.1**). New payoffs will be chosen at random as draws from from the same probability distribution used to generate the original payoffs. The payoff for each act will change independently of the others, so that p_c also represents the average proportion of payoffs that change in each round.

1.1.3 In the *spatial* case, the three demes share the same set of 100 behavioural acts, but the initial payoffs of each act are independently drawn from the same distribution across the demes. Subsequent changes in payoff are also independent in occurrence and magnitude across demes, although the rate of change, p_c, is the same. Thus an act could pay 4 in deme 1, 8 in deme 2, and 1 in deme 3. If that payoff changes in deme 1, this implies nothing about whether its value will change or has changed in the other demes. In any given simulation

[1] Question regarding these details can be directed to the organisers via email to Luke Rendell (ler4@st-andrews.ac.uk) or the discussion forum of the tournament Facebook page (search for 'Social Learning Strategies Tournament')
[2] If it is found that results are identical for shorter simulation runs then we may reduce this number for computational convenience.
[3] Precise details of the distribution are not given in order to encourage generality in submissions.

1.1.4 Each individual has a behavioural repertoire, containing a subset of the acts from the table specified above. Individuals are born naïve; they have an empty repertoire. Each individual's repertoire can subsequently contain only those acts, and knowledge about their payoffs, that are acquired through some form of learning (i.e. INNOVATE or OBSERVE; see below). Note that environmental change means that the payoff recorded for a given act relates to when the act was learned, and if the payoff for that act has subsequently changed (see **1.1.2** above), then the payoff level that the individual has recorded in its repertoire may be wrong.

1.2 Moves

1.2.1 Participants must specify a set of rules, henceforth a 'strategy', detailing which move an individual should use in each simulation round. In all simulations, three options are available:

1. INNOVATE (individual selects a new act at random from those outside its current repertoire, and learns that act and its payoff)
2. OBSERVE (individual learns the act(s) and estimated payoff(s) of one or more other individuals)
3. EXPLOIT (individual performs a specified act from its repertoire and reaps the payoff)

In simulations with the *cumulative* extension enabled, a fourth option is available, REFINE (**1.2.7**), where an individual specifies a behavioural act that it already knows, and improves it (increasing their refinement level for that act by 1) such that its payoff is increased.

1.2.2 INNOVATE is equivalent to trial-and-error learning, and does not guarantee an improvement in available payoffs. INNOVATE selects a new act at random from those acts not currently present in the individual's repertoire and adds that act and its exact payoff to the behavioural repertoire of the individual. If an individual already has the 100 possible acts in its repertoire, it gains no new act from playing INNOVATE. In the *cumulative* case, the new act is acquired with refinement level 0.

1.2.3 OBSERVE is equivalent to social learning. Individuals observe the act(s) and estimated payoff(s) being used by other individuals in the same round. This knowledge is then added to the observing individual's repertoire. Only those individuals playing EXPLOIT in the same deme as the observer are available to be copied, i.e. to become learning models. The sequence in which agents are selected to move does not affect the availability of models for copying – all agents are required to submit a move, and OBSERVE moves are processed subsequently, such that all agents playing EXPLOIT are potentially available to OBSERVE. The number of models observed is a parameter of the simulation, termed $n_{observe}$ (see **1.4.2**). These models are selected at random from those available, except under the *model-bias* extension, when agents can select whom to observe (see **1.2.5** below). If no individual plays EXPLOIT in that round then there are no models and nothing is learned. Similarly, if the number of models is less than $n_{observe}$, only the available models are observed. Note that it is also possible for an individual to OBSERVE an act already in its repertoire, in which case only the payoff recorded for that act is updated. In the *cumulative* case, the act is observed at the same level of refinement as demonstrated by the model, meaning the observer can acquire that level of refinement and its associated payoff increment, for that act. The observer does not however know whether it has observed a refined act or not, nor the associated level of refinement.

1.2.4 OBSERVE is error prone with regard to both act and payoff. Each of the $n_{observe}$ social learning events fails with a certain probability, and nothing is learned. The probability of failure is a parameter of the simulations, called $p_{copyFail}$ (see **1.4.3**). Where social learning fails, the individual receives no new behaviour or knowledge of its payoff. Furthermore, the payoff estimate returned will be a value drawn from a Poisson distribution with mean equal to the true payoff. This means larger values will be associated with larger errors.

1.2.5 In simulations with the *model-bias* extension, then individuals electing to play OBSERVE will subsequently be asked which of the available models they wish to copy, via the strategy's *observe_who* function. In simulations without this extension models are simply selected at random from the available pool, and this is also what happens if the strategy does not define an *observe_who* function. Thus it is up to entrants to decide if they want their strategy to make choices about whom to copy. The *observe_who* function is provided with the following information about every available model, giving various indexes of its performance: its age (in simulation rounds), its total payoff (the sum of all payoffs from EXPLOIT moves), the number of times it has been observed previously (irrespective of how that information was subsequently used), and the number of offspring it has had (see **1.3**). We assume this information, being social, is also error-prone, with error added in the same way as **1.2.5** – the returned value is drawn from a Poisson distribution with mean equal to the true value. This information is only made available once the decision to play OBSERVE has been made. The *observe_who* function must return the list of model information ranked in order of preference according to rules specified by the applicant, and the first $n_{observe}$ entries in this list become the learning models.

1.2.6 EXPLOIT is the only move that results in a direct payoff to the acting individual (EXPLOIT here does not mean that another individual is taken advantage of, only that an individual is exploiting its knowledge). Individuals playing EXPLOIT must specify which act they wish to deploy. An individual can only EXPLOIT acts it has previously learned. When an individual chooses to EXPLOIT an act, the payoff it receives is used to update the payoff recorded in its repertoire (that is, we assume an individual can, by performing an act, update its knowledge of how profitable that act is).

1.2.7 In *cumulative* simulations, the REFINE move is available, which enables an individual to invest time in improving a behavioural act that it already knows. Individuals playing REFINE must specify which act from their repertoire they wish to improve. The result is an increase by 1 of the refinement level that individual knows for the selected act. The resultant payoff available to that individual for that act is equal to the basic payoff defined by the environment (which can change; **1.1.2)**, plus an increment which is a function of the refinement level and is unaffected by basic payoff changes (see **1.4.5** for details of the function). Individuals will not know what refinement level they currently have for that act, only that they increase it by 1. They learn the new total payoff available for that act, without error.

1.3 Evolutionary dynamics: Lifespan, fitness, reproduction and migration

1.3.1 Evolutionary change will occur through a death-birth process. Individuals die at random, with probability of 0.02 per simulation round giving an expected lifespan of 50 rounds, and are replaced by the offspring of individuals selected to reproduce with probability proportional to their mean lifetime payoffs. The probability that individual *z* reproduces is $P_z / \sum_i P_i$, where P_z is the mean lifetime payoff of individual *z* (that is, the sum of its payoffs from playing EXPLOIT divided by the number of rounds *z* has been alive) and the denominator is the summed mean lifetime payoff of the population in that round. The mean lifetime payoff of an individual is unaffected by the number of offspring that it has produced.

1.3.2 Offspring are born behaviourally naïve in that they have no behavioural acts in their repertoire and no knowledge of payoffs. They do, however, inherit the learning strategy (i.e. tournament entry) of their parents, unless mutation occurs. Mutation will occur with probability 1/50, and when it does, the offspring will have a strategy randomly selected from the others in that simulation. These mutations are how other strategies will first arise in a population initially containing only a single strategy. Mutation will not occur in the last quarter of each *melee* simulation.

1.3.3 In the *spatial* case, a number of individuals, $n_{migrate}$, are selected at random from each deme, and then each reassigned to a randomly selected deme. Their repertoire is unaffected by migration – the agent will still know the same acts, at the same refinement levels in the *cumulative* case. They will not however know what the payoffs for those acts are in the new environment.

1.4 Simulation parameters and cumulative payoff function

1.4.1 Parameter p_c: The probability that the basic payoff of an act will change in a single simulation round. In the round-robin stage of the tournament, simulations will be run with a small number of p_c values, drawn from what we consider the biologically plausible range [0.001-0.4] In the *melee* stages, we will run simulations with more levels of p_c, drawn from the same range.

1.4.2 Parameter $n_{observe}$: The number of models copied by an agent playing OBSERVE. In the pairwise tournament phase $n_{observe}$ will take any one of a small number of fixed values from the range [1-10]. In the *melee* phases we will run simulations with $1 \leq n_{observe} \leq 10$.

1.4.3 Parameter $p_{copyFail}$: The probability that social learning will fail on any given copying event. In the pairwise phase this will be set to a single value, in the *melee* phases simulations will be run with values chosen from anywhere in the range between 0 and 0.5.

1.4.4 Parameter $n_{migrate}$: In *spatial* simulations only, this is the number of individuals chosen at random from each deme for migration. These are then reassigned to demes at random. In the pairwise stages, this will be set to a single fixed value, in the melee stages it will be chosen from anywhere in the range between 1 and 20.

1.4.5 Parameter r_{max}: In *cumulative* simulations only, this is the maximum allowed refinement level. A cumulative payoff function defines how much will be added to the basic payoff of a behavioural act (defined in the environment) for a given refinement level r. This level, r, can range from 0 for a newly innovated act, up to $r_{\max}$. The function defining this increment, i, for any given r, is

$$i = \left[\left(\frac{0.05}{1 - 0.95^{r_{\max}}} \sum_{j=1}^{r} 0.95^{r-j} \right) p_{\max} \right],$$

where the square brackets denote rounding to the nearest integer and $p_{\max}$ represents the maximum possible increment, defined as the mean of the basic payoff distribution multiplied by 50. This equation gives a diminishing returns function but still offers payoff increments well in excess of the expected mean of the basic payoffs (Figure 2). In the pairwise stages, r_{max} will be set to a single value, while in the melee stages it could be anywhere in the range 10-1000.

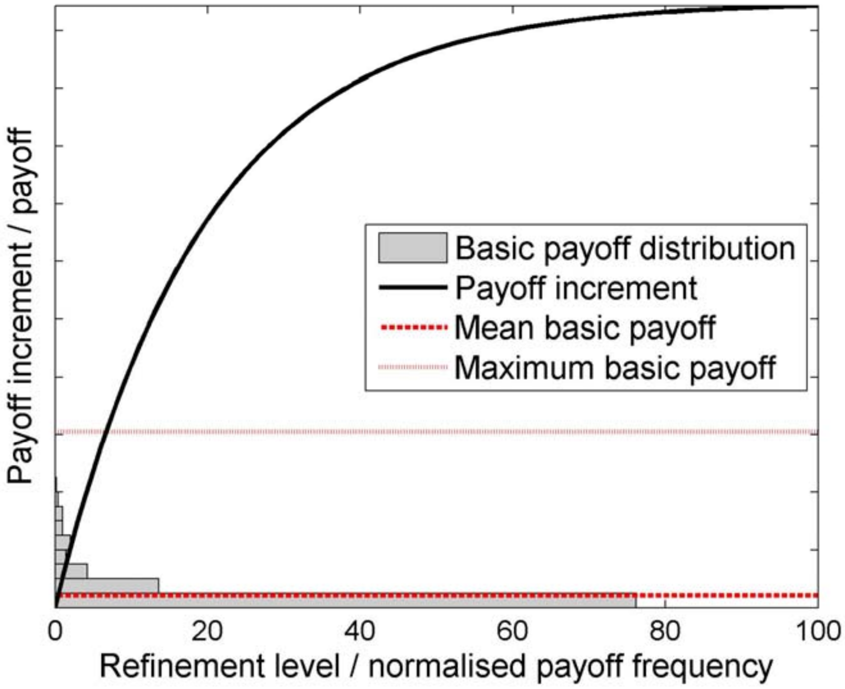

Figure 2: Relationship between refinement level and resultant payoff increment plotted along with a typical basic payoff distribution (here, r_{max} = 100). In *cumulative* simulations, the increment is added to the basic payoff. Absolute payoff values are deliberately omitted, and basic payoff distribution is indicative only.

2. Running the simulations

Details of how the simulations will run, and how scores will be recorded in each evaluation stage, are as follows:

2.1 *Stage 1: Single extension pairwise*

Strategies will take part in round-robin contests against all other strategies. A contest involves each strategy invading a population of the other strategy. In a given simulation, a population of the dominant strategy will be introduced, and run for 100 rounds to establish behavioural repertoires. At this point, mutation will be introduced, providing the second strategy the opportunity to invade. Simulations will then run for up to a further 10,000 rounds. Each pairwise contest will be repeated at least 5 times with strategy A as the invader and an equal number with strategy B as the invader, giving a minimum of 10 replicates. The mean frequencies of each strategy in the last quarter of each run (i.e. the last 2,500 rounds in a 10,000 round run) will be averaged over the repetitions. This average will then be recorded as the score of that strategy in that contest. Strategies will be assessed on their total score once every strategy has been tested against every other strategy. This process will be repeated three times, each with a different simulation extension (*model bias, cumulative,* or *spatial*) enabled in isolation.

2.2 *Stage 2: single extension melee*
Stage 2 will also be repeated once with each extension enabled. Simulations will start with an initial population consisting of individuals with a simple asocial learning strategy (INNOVATE once and then EXPLOIT on every subsequent move). Every time an individual reproduces, it has a 1/50 probability of mutating to a strategy chosen at random from the 10 top scoring strategies from the pairwise contests with the same extension enabled. However, there will be no mutation in the last quarter of the simulation so that mutation does not unduly influence results when strategies have similar fitnesses. After up to 10,000 rounds, the mean frequency of each strategy in the last quarter of the simulation will be recorded as the score for that strategy. We will run multiple simulations and vary across them all four parameters, and, in the *cumulative* case, the cumulative payoff function, as detailed in **1.4**. The strategy with the highest average score across all the simulations with a given extension enabled will be declared the winner for that extension and the entrant(s) will win a €5,000 prize. The exact number of conditions we test will depend on computational constraints.

2.3 *Stage 3: all extensions melee*
Finally, at least the top 5 scoring strategies from each *single extension melee*, together with up to 5 wildcards, will go forward to a final set of simulations, again with varying parameters and cumulative payoff functions, but with all three extensions simultaneously enabled. We define a 'wildcard' as any strategy that achieves a mean performance across all three contests combined that is within the top 5 of all entered strategies, but that has not been put forward for stage 3 by virtue of being a top-performer on any of the specialized contests. The inclusion of wildcards is intended to identify effective 'generalist' strategies. Again, the exact number of conditions we test will depend on computational constraints. The strategy with the highest average score across these simulations will be declared the overall tournament winner and the entrant(s) awarded the grand prize of €10,000. Hence a successful strategy is able to win up to €25,000 in prize money.

3. How to enter

3.1 Strategies will take the form of computer code functions that take the data specified below as arguments and return a decision on which move to play (as well as details such as which individuals to copy. in particular cases). An example strategy is given below. Entrants do not need knowledge of any programming language, as entries must be submitted as 'pseudocode' (that is, linguistic instructions breaking down how decisions are made into a series of mathematical and logical operations that can each be directly translated into a single line of computer code[4]). As the tournament will be run in Python (v2.7, base modules and Numpy available), entrants familiar with that language can of course submit Python code directly, and a strategy template is provided on the tournament website. However, even if an entry is submitted as Python code, a pseudocode version must also be provided, to facilitate debugging. In all cases the code should be straightforward to translate between formats. We provide in section **3.8** below an example strategy in both Python and pseudocode form, and refer to that strategy by line number in the following descriptions. Strategies must also be accompanied by a brief prose description of how they are intended to function.

3.2 The *move* function must return an integer number representing the individual's move in this round, Pseudocode entries can simply specify the intended move as either INNOVATE, OBSERVE, EXPLOIT or REFINE. These moves will be represented in computer code and agent history data by the numbers -1, 0, 1 and 2 respectively. REFINE is only available when the *cumulative* extension is enabled, and the *move* function is provided with a Boolean true-or-false input, called *canPlayRefine*, which informs it if this is the case (i.e. it is true if the *cumulative* extension is enabled, false if not). Similarly, the *move* function is also provided with inputs called *canChooseModel* and *multipleDemes* which are true if the *model bias* and *spatial* extensions, respectively, are enabled and false otherwise.

3.3 If the returned move is EXPLOIT or REFINE then the *move* function must also specify which behavioural act to perform or refine (as an integer value equal to one of the acts in the individual's behavioural repertoire). This act must be present in the individual's repertoire. If any individual tries to EXPLOIT or REFINE an act not in its repertoire, or play REFINE when the *cumulative* extension is not enabled, then it gets nothing for that round – no payoff and no addition to the behavioural repertoire. On the assumption that such attempts are mistakes in strategy algorithms, we will, for strategies submitted sufficiently before the deadline (see rules 8 and 11 below), attempt to contact the entrant(s) and invite them to revise their strategy, provided they do so before the entry deadline expires.

3.4 For the purposes of the tournament, organisms will be assumed to 'know' their own individual histories of behaviour and the fitness payoffs they received, so strategies can access this information. Each individual also has access to its own behavioural repertoire. Strategies will be provided with this information in the form of a *n x* 2 table, where *n* is the number of acts in the repertoire, the first column of the array represents the acts themselves and the second represents their payoffs. We assume that an individual can remember what it did over its lifetime, and how long it has been alive. Thus strategies will be provided with information on age, moves, acts exploited or learned, associated payoffs, and migration history (which will only be informative in the *spatial* case).

3.5 Strategies will receive the above knowledge in the form of three variables: `roundsAlive`, `myRepertoire` and `myHistory`. An individual that has survived 5 model rounds might receive the following data:

[4] For an example, see Mangel & Clark (2000) *Dynamic state variable models in ecology*. Oxford University Press, e.g. p55.

`roundsAlive = 5`	Number of previous rounds this individual has survived.
`currentDeme = 1`	The deme the agent is currently in
`cumulative = False`	The REFINE move is *not* available

`myRepertoire = {‘19’:3,`
`‘12’:4,`
`‘64’:9}`
	The individual’s behavioural repertoire, containing three acts: 19, 12, and 64 with, according to the individual’s current knowledge, payoffs of 3,4 and 9 respectively.

```
historyRounds     = ( 1, 2, 3, 4, 5)
historyMoves      = ( 0,-1, 0, 1, 1)
historyActs       = (12,19,64,64,64)
historyPayoffs    = ( 4, 3,10, 9, 9)
historyDemes      = ( 2, 2, 1, 1, 1)
```

The agent’s life so far; `historyRounds` is a list of rounds played since birth, `historyMoves` records the move played each round (**3.2**), `historyActs` is the act learned or exploited in each round, and `historyPayoffs` is a record of the new payoff learned (OBSERVE, INNOVATE, REFINE) or collected (EXPLOIT). Finally, `historyDemes` records which deme the agent was in on each round.

In this example case, $n_{observe}$ = 1. In its first model round, this individual played OBSERVE. As a result, it added act 12 to its repertoire and learned that this act returned a payoff of 4. In the second round it played INNOVATE, and added the act 19 with payoff 3 to its repertoire. In the third round the agent migrated (not under its control) from deme 2 to deme 1; it then played OBSERVE again and learned the act 64 with observed payoff 10. In rounds four and five, this individual played EXPLOIT, performed act 64, and received a payoff of 9. Note that its actual payoff received for act 64 was not exactly equal to the payoff learned for act 64 (when playing OBSERVE on the third round), because of error in social learning (see **1.2.4**).

Note also that in the case of newly born individuals, there will be no data – the repertoire and history will be empty and if your strategy uses these inputs, it should specify what to do in that case (and not crash!). The example strategy given below is robust to this as it specifically checks if `roundsAlive>1` (see **3.8**).

3.6 Note that in above case, $n_{observe} = 1$. If $n_{observe} = 3$, for example, then the history inputs might look like this:

```
historyRounds     = ( 1, 1, 1, 2, 3, 3, 3, 4, 5)
historyMoves      = ( 0, 0, 0,-1, 0, 0, 0, 1, 1)
historyActs       = (12,86,10,19,64,12,-1,64,64)
historyPayoffs    = ( 4, 6, 1, 3,10, 7,-1, 9, 9)
historyDemes      = ( 2, 2, 2, 2, 1, 1, 1, 1, 1)
```

Here, each OBSERVE move is represented by $n_{observe}$ (=3) entries in each history variable, highlighted in bold. On the first OBSERVE move (round 1), the individual observed acts 12, 86 and 10 with payoffs of 4,6 and 1 respectively. On the second OBSERVE move (round 3) it observed two acts, 64 and 12. Note that there are two estimates here for the payoff associated with act 12 (4 in round 1, 7 in round 3). These differences could be due to the error in observing payoffs associated with OBSERVE or the fact that the agent moved between demes between the two OBSERVE move, and the same act can have different payoffs in different demes. In the case that fewer than $n_{observe}$ individuals play EXPLOIT in the previous round, then some information returned by OBSERVE will be set to -1, as is shown for the underlined data from round 3 (-1 will also be returned if an individual that already has 100 acts in its repertoire plays INNOVATE). The behavioural repertoire for this individual would now contain acts 12, 86, 10 and 64 with payoffs 7,6,1 and 9 respectively.

3.7 There is no limit to the length of the function, but it cannot, on average, take more than 25 times as long as the example strategy, given in section **3.8**, to reach a decision. If, on completion of the tournament, this is found to be the case for your strategy, then it will not be eligible to win the tournament. However, if it proves to be an effective strategy, we may still discuss it in our reports of the tournament. Strategies are forbidden to access the disk or memory storage of the computer in any way beyond the information provided as input, so there is no way to store other information between rounds.

3.8 We provide below an example strategy to illustrate what is expected of entrants. This strategy is called ‘copy when payoff decreases’ (here given the function name ‘cwpd’). This example illustrates how strategies must be prepared to handle the start of an individual’s life, when it has no acts in its repertoire, and how strategies can change as individuals survive over several rounds.

Prose description:

This strategy always plays INNOVATE on the first round of life. On the second round, it will EXPLOIT the act innovated on the first round. Thereafter it continues to EXPLOIT that act unless the returned payoff drops below the average payoff from all the EXPLOIT moves in its life,, at which point it plays OBSERVE. After playing OBSERVE, it always returns to EXPLOIT for at least one move. When playing OBSERVE in the “model bias” scenario it will preferentially learn from older models. If the ‘cumulative’ extension is enabled, then it will play REFINE with probability 1/20 whenever it would have played EXPLOIT.

Pseudocode version:

copyOldestWhenPayoffsDecrease

```
move:
1. If roundsAlive=0 then INNOVATE (move=-1)
2. If roundsAlive=1 then EXPLOIT with behaviour learned in first
   round (move = first value in myR)
3. If roundsAlive>1 then:
   4. Calculate my average payoff when EXPLOITing, call it
      myMeanPayoff
   5. Find out when I last EXPLOITed, and store the payoff from that
      EXPLOIT as lastPayoff
   6. Find what my last move was, store as lastMove
   7. If lastPayoff<myMeanPayoff and lastMove<>OBSERVE then OBSERVE,
      except if cumulative=True then REFINE current best act with
      probability 1/20
   8. Otherwise EXPLOIT the act with highest current known payoff
observe_who:
1. Rank available models by age, oldest first
```

Python version (text in green are comments to aid interpretation):

```
from moves import * #bring in standard names for moves
#this means INNOVATE, OBSERVE, EXPLOIT and REFINE can be used instead of -1,0,1,2
#and that AGE, TOTAL_PAY, TIMES_COPIED and N_OFFSPRING can be used to index into
                                                       exploiterData
import random,math

def move(roundsAlive, repertoire, historyRounds, historyMoves, historyActs,
historyPayoffs, historyDemes, currentDeme, canChooseModel, canPlayRefine,
multipleDemes):
'roundsAlive, currentDeme are integers, history* are tuples of integers'
'repertoire is a dictionary with keys=behavioural acts and values=payoffs'
'canChooseModel, canPlayRefine, multipleDemes are boolean, indicating whether:'
'observe_who will be called (i.e. model bias), REFINE is available (i.e.
cumulative), and multiple demes (i.e. spatial) respectively'
'This function MUST return a tuple in the form (MOVE,ACT) if MOVE is EXPLOIT or
REFINE, or (MOVE,) if MOVE is INNOVATE or OBSERVE'

    if roundsAlive>1: #if this isn't my first or second round
        myMeanPayoff = sum([p for i,p in enumerate(historyPayoffs) if
           historyMoves[i]==EXPLOIT])/float(historyMoves.count(EXPLOIT))
           #calculate mean payoff from playing exploit
        lastPayoff = historyPayoffs[len(historyMoves)-1-historyMoves[::-
           1].index(EXPLOIT)] #get last payoff from exploit
        lastMove = historyMoves[-1] #get last move
        if lastMove==OBSERVE or lastPayoff>=myMeanPayoff: #if lastMove was
               observe or lastPayoff at least as good as myMeanPayoff
            if random.random()<0.05 and canChooseModel: #if simulation allows
                  refinement
                return (REFINE,max(repertoire, key=repertoire.get)) #then REFINE
                    best known act with probability 1/20
            else:
                return (EXPLOIT,max(repertoire, key=repertoire.get)) #otherwise
                    EXPLOIT best known act
        else:
            return (OBSERVE,) #if payoffs have dropped then OBSERVE
    elif roundsAlive>0: #if this is my second round
        return (EXPLOIT,repertoire.keys()[0]) #then EXPLOIT the act innovated on
           the first round
    else: #otherwise this must be the first round
        return (INNOVATE,)
```

```
def observe_who(exploiterData):
'This function MUST return the given list of tuples, exploiterData, sorted by
preference for copying.'
'Data given for each agent are (index in this list,age,total accrued
payoff,number of times copied,number of offpsring)'
'All values except index have error applied'
    return sorted(exploiterData,key=lambda x:x[AGE],reverse=True) #copy oldest
```

3.9 From the above, it can be seen that entrants can engage with all, any, or none of the three extensions. To engage with the *model bias* extension, a strategy should specify an `observe_who` function; if it does not, then models will be selected at random from those available. To engage with the *cumulative* extension, a strategy can choose to play REFINE, but it should check whether `cumulative` is true, and provide an alternative move if it is not. If it does not then entered strategies will simply never REFINE. Finally, to engage with the *spatial* extension, a strategy can use the information provided by `historyDemes` and `currentDeme` in any way it wishes.

3.10 Entries should be emailed as attachments to Luke Rendell <ler4@st-andrews.ac.uk>. Please make sure you read and accept the rules below before submitting your entry. We recommend that you use the entry form on the tournament website to make sure all the required information is submitted.

4. Tournament Rules

1. Entry into the tournament must be accompanied by explicit acceptance of these rules.

2. The decisions of the committee shall in all cases be final and binding.

3. Anyone may enter the tournament, with the exception of current members of Kevin Laland's research group, and members of the committee. Students/postdocs of other committee members are permitted to enter, but the committee will not be informed of entrant's identities if they are asked to adjudicate specific issues.

4. Entrants may be single individuals, or collaborative groups. In the latter case, groups must select a corresponding entrant who will be the sole point of contact for the tournament organisers and the only person with whom the organisers will discuss that entry. Groups must also provide a list of the group members. Entrants may submit only one strategy, and individuals may only participate in one group entry.

5. Only entries received by 1700 GMT on the closing date, February 28th 2012, will be accepted, and no further modification of entries is permitted after this date. Entrants are *strongly* advised to submit their strategies well before this time so that the organisers can check the code and inform entrants of any problems before the final closing date (see 9 and 12).

6. All entrants agree to the *content* of their submission being made public as part of the communication of this research exercise, although entrants can choose not to have their *name* associated with their entry.

7. There is no limit to the length of the function, but it cannot, on average, take more than 25 times as long as the example strategy, given in section **3.8**, to reach a decision. If, on completion of the pair-wise tournament, this is found to be the case for your strategy, then it will not be eligible to win the tournament. However, if it proves to be an effective strategy, we may still discuss it in our reports of the tournament.

8. Your strategy cannot access the disk or memory storage of the computer in any way beyond the information provided as input. The organizers reserve the right to disqualify strategies that are deemed not in the spirit of the contest.

9. Strategies playing EXPLOIT or REFINE must specify which act to use from their repertoire. This act must be present in the individual's repertoire. If any strategy returns acts not in the repertoire then on the assumption that such attempts are mistakes in strategy algorithms, provided the strategy submitted sufficiently before the deadline (see 11 below) we will attempt to contact the entrant(s) and invite them to revise their strategy, provided they do so before the entry deadline expires.

10. We reserve the right to edit code for computational efficiency, but we will notify entrants if this occurs and they will be given the opportunity to check that the operation of their strategies has not been compromised.

11. Strategies *must* be accompanied by both brief prose description of how they are intended to function and, if submitted as computer code, a ‘pseudocode’ version.

12. Entrants must be prepared to enter into a reasonable dialogue with the organisers to remove ambiguities from the entered strategy for the purposes of coding the simulations and improving computation efficiency. We will endeavour to inform entrants if their strategies do not function correctly. If a strategy is deemed inadmissible prior to the closing date, entrants will be informed forthwith and given the opportunity to revise their submission. Strategies deemed inadmissible after the closing date will be disqualified. We guarantee to pre-test any strategies submitted up to one month before the closing date; we can give no such guarantee for strategies submitted after this time, although we will endeavour to do so.

13. If a strategy is submitted that, in the opinion of the organisers, is so similar to one already submitted as to be reasonably considered identical, then the first submission will take precedence and the submitter of the identical strategy will be informed that their entry is ineligible. The submitter will however be eligible to revise and resubmit their entry, provided that they do so prior to the closing date.

14. Any strategy that, in the opinion of the organisers, has been designed so as in any way to recognise and specifically help other entered strategies at their own expense will be disqualified and the authors of the strategy will be given no further opportunity to enter a modified strategy. This rule is essential to preserve the evolutionary validity of the tournament.

15. In the event of equivalences in performance, either at the cut off point for going forward to the next stage or amongst the stage winners, the ‘tied’ strategies will be submitted to further tests under varied simulation conditions as deemed appropriate by the organising committee. In the case of the top-performing strategies for stages 2 or 3, if the committee judges that the tied strategies do indeed have equal merit, then they may decide at their discretion to share the prize between the tied entrants.

16. In the event that the number of submitted strategies renders a complete set of pairwise contests computationally unfeasible, we reserve the right to use a different system to select which strategies move forward to the *melee* stage, for example by splitting the strategies into randomly assigned groups from which winners will be selected to go forward to the *melee* stage.

17. The winning entrant in each *single extension melee* will receive a cash prize of €5,000, and the winner of the *all extension melee* will receive €10,000. There is no restriction on the number of prizes a single entry can win. In the first tournament, the winners were invited to co-author a paper reporting on the

contest, and the organizers would look favourably on a similar outcome for this tournament, although they also reserve the right to produce the paper without the entrant's participation and to judge authorship merits at their discretion.

References

Axelrod, R. 1980. Effective choice in the Prisoner's Dilemma. J. Conflict Resolution 24:3-25.

Barnard, C. J., and R. M. Sibly. 1981. Producers and scroungers – a general-model and its application to captive flocks of house sparrows. Anim. Behav. 29:543–550.

Boyd, R., and P. J. Richerson. 1985. Culture and the Evolutionary Process. Chicago University Press, Chicago.

Boyd, R., and P. J. Richerson. 1995. Why does culture increase human adaptability? Ethol. Sociobiol. 16:123-143.

Enquist, M., K. Eriksson, and S. Ghirlanda. 2007. Critical social learning: a solution to Rogers' paradox of non-adaptive culture. Am. Anthropol. 109:727-734.

Feldman, M. W., K. Aoki, and J. Kumm. 1996. Individual versus social learning: Evolutionary analysis in a fluctuating environment. Anthropol. Sci. 104:209-231.

Galef Jr., B. G. 1995. Why behaviour patterns that animals learn socially are locally adaptive. Anim. Behav. 49:1325-1334.

Giraldeau, L.-A., T. J. Valone, and J. J. Templeton. 2002. Potential disadvantages of using socially acquired information. Phil. Trans. R. Soc. B 357:1559-1566.

Henrich, J., and R. McElreath. 2003. The evolution of cultural evolution. Evol. Anthropol. 12:123-135.

Hoppitt, W., and K. N. Laland. 2008. Social processes influencing learning in animals: A review of the evidence. Adv. Study Behav. 38:105-165.

Laland, K. N. 2004. Social learning strategies. Learn. Behav. 32:4-14.

Leadbeater, E., and L. Chittka. 2007. Social learning in insects — From miniature brains to consensus building. Curr. Biol. 17:R703-R713.

Rendell, L., R. Boyd, D. Cownden, M. Enquist, K. Eriksson, M. W. Feldman, L. Fogarty, S. Ghirlanda, T. Lillicrap, and K. N. Laland. 2010. Why Copy Others? Insights from the Social Learning Strategies Tournament. Science 328:208-213.

Rendell, L., R. Boyd, M. Enquist, M. W. Feldman, L. Fogarty, and K. N. Laland. 2011. How copying affects the amount, evenness and persistence of cultural knowledge: insights from the social learning strategies tournament. Phil. Trans. R. Soc. B 366:1118-1128.

Rogers, A. 1988. Does biology constrain culture? Am. Anthropol. 90:819-813.

Schlag, K. H. 1998. Why imitate, and if so, how? J. Econ. Theory 78:130-156.