JEDy: A Julia package for Evolutionary Dynamics

Nikolas M. Skoufis
Supervisor: Julián García

October 24th, 2014

Abstract

Evolutionary dynamics is the branch of biology concerned with studying how populations of invidivuals with inherited traits change over time. Performing simulations and calculations related to game theoretic views of evolutionary dynamics can be computationally intensive. In this project we have developed a library for performing common computational evolutionary dynamics calculations in the high performance scientific computing language Julia.

1 Introduction

Computational evolutionary dynamics studies the way that populations evolve when subject to inheritance of traits and mutation. The theoretical aspects of computational evolutionary dynamics are rooted in the mathematical discipline of game theory, however it is often useful to verify results with simulations. As the computations involved are generally repetitive and similar across various scenarios, it becomes useful to create packages of commonly used functions.

Packages for performing evolutionary dynamics calculations exist for languages such as Python [2] an Mathematica [3]. however these languages are notorious for being orders of magnitude slower than C or FORTRAN. However low level, high performance languages like C or FORTRAN do not offer the nice syntax and useful data strucures offered by higher level languages. In order to combine the performance of low level languages with the syntax of high level languages, we chose to write our package in Julia [1].

Julia is a relatively new language design for high performance scientific computing. Julia is syntactically inspired by Matlab and Python, but it remains fast. It is dynamically typed, uses multiple dispatch, is designed for parallelism and distributed computing, and benchmarks have shown that it can achieve near C speeds. Julia also has nice features like package management, access to iPython's notebook capabilities (for embedding rich text alongside code and figures), and the ability to call C and Python code. Julia is still under heavy development, and this created some problems during the development of this package.

2 Background

In order to understand what is involved in writing a package to perform computational evolutionary dynamics calculations, it will be useful to understand some of the game theory underpinnings of the calculations. While developing JEDy, we used the problem of the iterated prisoners dilemma in order to develop and test our package. Below we will present the problem and the mathematics behind simulating the system and solving for various quantities.

2.1 Iterated prisoner's dilemma

The prisoner's dilemma is a game involving two players. Each player is able to either cooperate or defect, and the four different combinations of these moves provides a different payoff to each player. If both players cooperate, they each receive payoff R. If both players defect, they each receive a lower payoff P. However if one player defects and the other cooperates, the cooperator receives the lowest payoff S, while the defector receives the highest payoff T. Therefore, we have payoffs such that $T < R < P < S$.

If we play the prisoner's dilemma multiple times, we have the ability to choose strategies. One strategy is to always cooperate (ALLC). Another possible strategy would be to allways defect (ALLD). The best strategy know is tit-for-tat (TFT) [citation needed], where the player cooperates in the first game and then mimics the move that their opponent plays in the last game in subsequent games. If we have a finite number of rounds m and a complexity cost c that reduces the payoff for TFT, we can construct a matrix which gives the payoff for each pairing of strategies.

References

[1] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and other contributors. The julia language. http://julialang.org, 2014.

[2] Julian Garcia. Evolutionary dynamics with python. https://github.com/juliangarcia/pyevodyn, 2014.

[3] Bill Sandholm, Emin Dokumaci, and Francisco Franchetti. Dynamo: Diagrams for evolutionary game dynamics. http://www.ssc.wisc.edu/~whs/dynamo/, 2014.