MBS Technical Report 15-01

Colorsims: A python package for evolving linguistic color naming conventions within a population of simulated agents

Sean Tauber*

Institute for Mathematical Behavioral Sciences,
University of California, Irvine

Abstract

Colorsims is a python package for simulating the cultural evolution of linguistic color naming conventions. Simulations are modular, allowing the user to explore the effects of varying different assumptions such as the color space, environment, agent learning mechanisms, population size, social network structure, and evolutionary dynamics. Utilities for storing and visualizing simulation data are included.

1 Introduction

Colorsims is a python package developed for simulating cultural evolution of linguistic color naming conventions. Several previous studies [8, 10, 2] were based on early versions of this software. Colorsims in its current form has been used to facilitate undergraduate research projects conducted as part of the Multidisciplinary Design Program (MDP)—which is facilitated by the Undergraduate Research Opportunities Program (UROP) and Calit2—at the University of California, Irvine. The software provides a straightforward implement of simulations that can replicate previous findings [8, 5, 4, 3, 6, 7]

*Email: stauber@uci.edu

about the evolution of linguistic color naming conventions within populations of agents. It facilitates the exploration of novel questions using a modular design that maps key theoretical properties to configurable aspects of the simulation. For example, by adjusting configuration parameters the user can easily modify components of the simulation such as the color space, environmental contingencies, evolutionary dynamics, agent learning mechanisms, agent population size and social network structure.

2 Features

The package uses a modular, object oriented design in which key modules and classes intuitively correspond to theoretical abstractions from the literature on the evolution of color naming conventions [8, 5, 4, 3, 6, 7]. In this section I outline the abstractions incorporated in colorsims and the ways in which they can be implemented and modified by the user.

2.1 Color space

The default color space provides an abstraction for a discrete set of color chips on a wheel. The user specifies the number of chips and the sampling distribution—which determines the probability, for each chip, that it is chosen when randomly sampling a chip from the color space.

Methods are provided for several useful functions such as random sampling of color chips and computing the shortest distance between two particular chips on the wheel.

2.2 Agents

Agents provide an abstraction for individuals who play a color-naming game while interacting with the environment and other agents..

The default agent is a *reinforcement learning* (RL) agent. An RL agent has a vocabulary of words—linguistic terms that are used to name colors—and a probabilistic strategy for using these words to name each color chip in the color space. When presented with a color chip, the RL agent uses its strategy to generate a name for the chip. Consequently, the agent receives reinforcement or punishment from the environment which causes it to increase

or decrease the probability of using the chosen name for that color chip in the future.

The user can adjust several aspects of the agent that affect the reinforcement process, such as the relative impact that each reinforcement or punishment has on the agent's naming strategy.

2.3 Populations

Populations consist of a collection of agents and a social network structure defining the connections between agents. The user specifies the size (number of agents) in the population and chooses a network structure.

Networks that are available by default are complete networks, where every agent is connected to every other agent, and grid layouts, where agents are only connected to their closest geospatial neighbors. It is fairly straightforward to implement new populations with custom network structures by overriding the network initialization method of the default population class. Networks are undirected graph objects from the popular networkx [9] python package which contains many predefined network structures. Networkx also provides a simple interface for building custom networks by defining the edges between nodes.

There are several utility methods that allow the user to save and load populations from disk, and to plot and save visual representations of a population.

2.4 Environment and dynamics

Users can customize the dynamics of the interactions between agents, the environment and other agents. For example, the criteria determining whether agents receive reinforcement or punishment from the environment can be set in different ways. Similarly, the user can define different types of social learning between agents.

2.5 Simulation utilities

There are some basic utilities for managing and visualizing simulation data. Simulations can be configured to run for a specified number of iterations and automatically save the state of the population at regular intervals. There are visualization tools that generate a series of images from the saved simulation

data. It is also possible to automatically build a movie from this series of images.

3 Availability

3.1 Operating system

The package has been tested on Mac OS X and Windows based systems. In principle, it should run on any system on which python 2.x is installed and all dependencies (i.e. external python libraries) are properly configured.

3.2 Programming language

The package was developed with python 2.7.9 and should be compatible in general with any python 2.7.x installation. Some portions of the code are incompatible with python 3.x.

3.3 Dependencies

The core functionality depends heavily on the following external packages: Numpy [11], matplotlib [1] and networkx [9]. These packages must be properly installed and configured in your python distribution. I highly recommend the anaconda python distribution from Continuum Analytics[1] which includes a number of preconfigured packages for scientific computing—including all three of the core dependencies for colorsims.

3.4 Code

The code is available by request of the author.

Author notes

Funding provided by the University of California Pacific Rim Research Program, 2010-2015 (Jameson, PI); the National Science Foundation 2014-2017

[1] http://continuum.io/

(award: SMA-1416907, K.A. Jameson, PI); and the Airforce Office of Scientific Research 2012-2015 (award: FA9550-13-1-0012, L. Narens, PI). The opinions expressed in this paper are those of the authors and do not reflect the opinions of any of the above funding organizations.

References

[1] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science and engineering*, 9(3):90–95, 2007.

[2] K.A. Jameson, N.L. Komarova, S. Tauber, and L. Narens. New results on simulated color categorization behaviors using realistic perceptual models, heterogeneous observers and pragmatic communication constraints., 2011. Presentation at The 17th annual meeting of the Cognitive Science Association for Interdisciplinary Learning. August 2011. Hood River Valley, OR.

[3] Kimberly A Jameson and Natalia L Komarova. Evolutionary models of color categorization. i. population categorization systems based on normal and dichromat observers. *JOSA A*, 26(6):1414–1423, 2009.

[4] Kimberly A Jameson and Natalia L Komarova. Evolutionary models of color categorization. ii. realistic observer models and population heterogeneity. *JOSA A*, 26(6):1424–1436, 2009.

[5] Kimberly A Jameson and Natalia L Komarova. Evolutionary models of color categorization based on realistic observer models and population heterogeneity. *Journal of Vision*, 10(15):26–26, 2010.

[6] Natalia L Komarova and Kimberly A Jameson. Population heterogeneity and color stimulus heterogeneity in agent-based color categorization. *Journal of theoretical Biology*, 253(4):680–700, 2008.

[7] Natalia L Komarova, Kimberly A Jameson, and Louis Narens. Evolutionary models of color categorization based on discrimination. *Journal of Mathematical Psychology*, 51(6):359–382, 2007.

[8] L. Narens, K.A. Jameson, N.L. Komarova, and S. Tauber. Language, categorization, and convention. *Advances in Complex Systems*, 15(03n04), 2012.

[9] Daniel A Schult and PJ Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16, 2008.

[10] S. Tauber, L. Narens, and K.A. Jameson. Evolutionary models of color categorization on networks, 2011. Talk at the annual meeting of the American Society for Mathematical Psychology, Tufts University, Medford, MA, July 2011.

[11] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.