IMBS TECHNICAL REPORT 18-02

ColorSims 2.0: An extension to the python package for evolving linguistic color naming conventions applied to a population of agents

Maryam Gooyabadi[1,*] and Kirbi Joe[1,*]

[1]*Institute for Mathematical Behavioral Sciences, University of California, Irvine*

Abstract

ColorSims 2.0 is an extension to the existing python package ColorSims [14] which includes notable updates and additions to the original package. ColorSims/ColorSims 2.0 is a python package for simulating the cultural evolution of linguistic color naming conventions. The package can be initialized with random agents or population data (i.e. World Color Survey). The components of the package are modular, allowing the user to vary them independently. Implemented parameters include: dimensions of the color space, population size, social network structure, and agent learning mechanisms (i.e. reinforcement learning, updating) within the evolutionary dynamics in addition to on-board utilities for storing data and visualizing simulation results.

Key words: Simulations, Color Categorization, Evolutionary Dynamics, Naming Systems, World Color Survey, Learning, Python

---

* Corresponding authors: Joe, K. (joek@uci.edu) and Gooyabadi, M. (mgooyaba@uci.edu)

1 Introduction

ColorSims 2.0 is an extension of the existing Python package ColorSims [14] that allows users to explore novel questions in the area of *cultural evolution of linguistic color naming conventions/systems*. It is a modular python package initialized by five properties that can be used to configure simulation investigations on color categorization system evolutionary dynamics:

1. Dimensions of the color space
2. Population size
3. Social network structure
4. Agent learning mechanisms (ie. updating, reinforcement learning)
5. Agent type (i.e. initialized randomly or using real observer population data).

This report details new additions found in ColorSims 2.0 including: extending investigations to three-dimensional color space, incorporation of a Stability Measure to run simulations to completion, and processes to initialize agents with World Color Survey (WCS) data.

The above mentioned improvements build upon the originally implemented forms of learning mechanisms within the evolutionary dynamic, namely reinforcement learning (i.e. student-teacher game) and updating, and the network structure capabilities seen in earlier ColorSims implementations [14].

1.1 ColorSims - original package

Several previous studies [10, 15, 7] were based on earlier ColorSims versions. These studies used randomized agents over a one-dimensional color space, in which ColorSims was additionally used as a teaching tool to facilitate undergraduate research projects conducted as part of the Multidisciplinary Design Program (MDP)—facilitated by the Undergraduate Research Opportunities Program (UROP) and Calit2—at the University of California, Irvine.

2 Features of ColorSims 2.0

2.1 Higher-dimensional Color Spaces

In ColorSims 2.0, the default color space is a two-dimensional array of colored stimuli, an update to the one-dimensional space of ColorSims [14]. This array is a subset of the standardized set of 330 (320 chromatic, 10 achromatic) Munsell color chips, which derived from a Mercator projection of a three-dimensional Munsell Book of Color perceptual color space. The set of stimuli used in the simulations is identical to the chromatic component of the stimulus palette used in the World Color Survey [11], as seen in Figure 1.

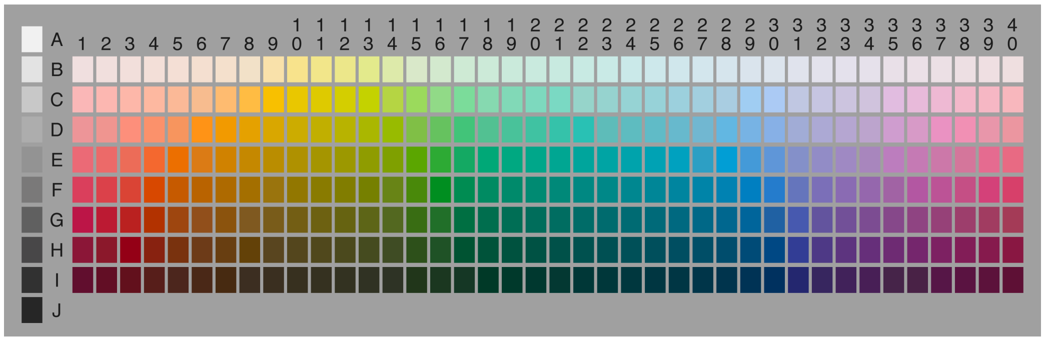


Figure 1: Stimulus array used in the World Color Survey. The 10 achromatic chips on the left of the array were omitted from the stimulus set used in ColorSims 2.0.

For the purposes of employing a standard color difference metric, each of the chips shown in Figure 1 is mapped to its corresponding coordinates in the three-dimensional CIELUV color model, which essentially allows a principled assessment of agents color discrimination judgments based on a standardized uniform perceptual color space. CIELUV was developed by the International Commission on Illumination (CIE) in 1976 with the aim of creating a perceptually uniform space [13]. When presented such stimuli in evolutionary game situations, agents ostensibly "perceive the colors" as 3-tuples (L^*, u^*, v^*), the coordinates of the colored stimuli in CIELUV space. While the set of stimuli is constricted to two-dimensions as shown in Figure 1, the underlying space that agents perceive and judge is represented as three-dimensional.

Thus, while existing simulation studies have primarily utilized a one-dimensional hue space called a hue circle (i.e. a discrete array of color chips arranged according to just-noticeable-differences) [14], ColorSims 2.0 implements a three-dimensional perceptual color space in order to more realistically approximate how humans represent and perceive color similarity. The CIELUV color model is appropriate for use here since its approximate perceptual metric permits computation of color difference, or Delta-E, by employing the fact that physical distances between colors map to similar distances in our perception. Similarly, by using a perceptually uniform space, agents can use distances between chips in CIELUV space as a basis for judgments of color discrimination when engaged in naming game scenarios used to evolve the population naming conventions of certain chips (see *Section 2.2* for more information on color discrimination measures).

In addition to the above mentioned color metric issues, expanding the stimulus space to Figure 1's two-dimensional array of Munsell color chips also permits the novel initialization of agent populations based on real observer data collected from the World Color Survey. This is possible because Figure 1's array was the stimulus set used in the World Color Survey to elicit participant color categorizations. Thus, by using a subset of the WCS stimulus set, we are able to evaluate questions relating to what the stability of observed WCS categorizations were at the time of collection.

2.2 Color Discrimination Measure: k-sim

On a one-dimensional color space, discrimination judgments of color naming are made according to a measure called *k-sim*. K-sim is defined to be the minimum number of chips at which it becomes important, for pragmatic purposes, to treat two color chips as different color categories [8]. If two chips are within k-sim, they are considered to be of the same category. If two chips are outside k-sim, they are considered to be of different categories.

In the one-dimensional case, distance between chips i and j is defined to be the number of adjacent color chips between chips i and j [14]. However, counting the number of chips between two color chips is not an intuitive measure of distance when the stimulus set is extended to two dimensions. Therefore, we redefine distance based on perceptual distances instead of the number of chips. Since the CIELUV space aims to be perceptually uniform, calculating color difference as distances in the physical space will in essence map to similar distances in human perception. Thus, distances between stimuli in the two-dimensional color space are calculated using the standard Euclidean distance metric:

$$d = \sqrt{(L_1^* - L_2^*)^2 + (u_1^* - u_2^*)^2 + (v_1^* - v_2^*)^2} \tag{1}$$

where $d =$ distance between two chips, $(L_i^*, u_i^*, v_i^*) =$ coordinates of chip i in CIELUV space. K-sim is then redefined in terms of this metric, instead of a number of chips. The interpretation of k-sim remains the same as in the one-dimensional case.

2.3 Evaluating stability of solution

One goal of these simulation investigations is to study how naming systems arise and evolve over time. A naming system arises when a population converges on a naming system as a whole, or when there exists a high level of agreement in category names and partitions amongst all individuals in the population. Using ColorSims 2.0, a solution is considered to have reached convergence when agent error is minimized, resulting in an optimum number of categories. Typically, these categories will be of roughly equal size. Though error is minimized, complete agreement is never reached since errors will continue to occur at the color boundaries. Hence, the convergent solution is a non-Nash equilibrium. Instead, the solution is said to be stable if the naming system stays the same for a very long period of time.

The two measures needed to determine whether a population's color naming system is stable are:

1. The level of lexicon agreement across the whole population
2. The percent change of that global agreement level across time

Global lexicon agreement on round r of the game (A_r) is defined as follows:

$$A_r = \sum_{i=1}^{320} \frac{P_{i,\alpha}}{N} \tag{2}$$

where r is the game round number, i is the chip number, α is the most frequently used name for chip i, N is population size, and $P_{i,\alpha}$ is number of agents who assign name α to chip i.

The global agreement level is then used to calculate the *Stability Measure*, defined as follows:

$$S_r = \frac{A_r - A_{r-10,000}}{A_r + A_{r-10,000}} \tag{3}$$

Global agreement (A_r) is calculated every 10,000 rounds, so S_r is interpreted as the change in agreement level between rounds r and $(r - 10,000)$, or in other words, the change in agreement level between each "snapshot" of the population.

The simulation marks an r^*, which is the round number at which the stability measure first falls within some predetermined "stability range" (default range is (-0.00175, 0.00175), determined empirically). The simulation will stop if the solution stays stable—S_r stays within the stability range—for another r^* rounds. That is, a naming system is considered to have converged to a stable solution if the system is stable for r^* rounds, resulting in an overall total of $2r^*$ simulation rounds.

We define the S_r in order to understand at which round r the population has converged on a naming system. In the original ColorSims, the number of rounds was a parameter given to the simulation. For simulations with "large" parameters, such as populations with many agents or stimulus sets with many chips, determining how many rounds to run the simulation was an educated guess based on trial and error. However, the naming system at the end of those predetermined number of rounds was not guaranteed to be stable. However, S_r ensures two things: firstly, the system will run for as long as it needs to reach convergence, as determined from S_r falling within a stability range; Secondly, we can check that the solution remains stable over a long period of time, as determined by r^*.

2.4 Initializing Population with Real Observer Data

The agents in the simulation can be initialized either by random chip-name assignment or with participant data from the World Color Survey. Random initialization allows evolution to proceed *de novo*, and a WCS language initialization allows the presumed evolution to precede to stability. Populations in the WCS simulations are taken to be the ∼25 people who participated in the WCS for a given language. Each agent's probability matrix for determining naming strategy is initialized to be a single participant's responses from the chip naming task in the World Color Survey.

By initializing the simulations with agents modeled after WCS participants, we can assess various aspects of a language's current—at the time of data collection—color categorization by allowing the solution to evolve over time. We can ask questions of the existing color naming systems, such as: (i) Does a stable, shared meaning system emerge? (ii) Are WCS society lexical categories that are used only by a proportion of each observed sample found to “drop out” from stable system solutions, and are new categories that are minimally represented in the human data shown to ultimately develop as robust categories? (iii) Do languages which are “endangered” by EGIDS standards (www.ethnologue.com) stabilize in different ways than languages that are “developing”? While these simulations do not provide actual predictions as to how naming systems shared by humans might evolve, they are useful in removing uncertainty from individual solutions. By allowing members of the society to engage in communication interactions of a fixed kind, with static pragmatic pressures, we can conduct comparative analyses across human systems.

Data from the WCS used to initialize agents is included in the ColorSims 2.0 package. In the package download there is a folder labeled “WCSParticipantData” which contains participant data for all languages, in the correct format, needed for the program.

2.5 Language constancy

The language constancy measure (L_w) allows us to calculate how stable or fixed a given language is over the duration of a simulation. This can be used to evaluate the constancy of color naming systems like those found in the World Color Survey. In calculating language constancy, we consider the distance between the language’s initial WCS data and the naming solution achieved following a WCS simulation, represented as the difference between the initial and final global agreement levels. We hypothesize that if a population were to participate in interactions of a specified nature exclusively within their community, a fairly high consensus about their color categorization should arise. Therefore, the solution that manifests from the simulation is considered to be an “exemplary” categorization, or in other words, the shared categorization that arises given repeated interaction. If, after running a language through a WCS simulation, the language’s naming system has not changed much, we say that the language has high language constancy. Alternatively, if the solution found at the end of a simulation is notably different from the initial system (i.e. more than one standard deviation (SD) away from the mean in the positive direction), then we say that the language has low language constancy. In other words, any L_w within one SD from the mean is considered moderately constant, below one SD is considered highly constant, and above one SD is considered lowly constant. The measure L_w is defined as follows:

$$L_w = \frac{|A_0 - A_{\bar{r}}|}{320} \tag{4}$$

where w is the WCS language number, $\bar{r}$ is total number of simulation rounds, A_0 is the initial global agreement level, $A_{\bar{r}}$ is the global agreement level at the end of the simulation, and 320 is the total number of color chips. L_w can take on values in (0, 1) (mathematically, $L_w \in [0, 1]$ but due to practical constraints, achieving the end points is highly unlikely, if not impossible). As the value of L_w increases, the level of language constancy decreases (e.g. a language that changed minimally between their initial and ending categorizations would have high language constancy with $L_w \approx 0$).

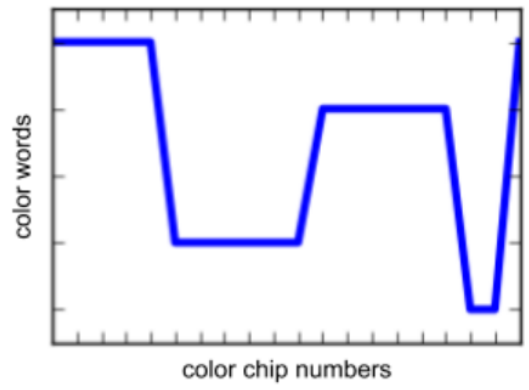
To test measure robustness, we defined another variable which captured population-wide agreement level at a specific round r called *pairwise agent comparison*, denoted C_r (See *Appendix A* for definition and formula). To calculate language constancy using pairwise agent comparison, the A_is in Formula 4 were substituted with C_is. Language constancy results were consistent regardless of which measure of agreement was used—global agreement (A_r) or pairwise agent comparison (C_r) (see *Section 3.2*).

2.6 Simulation utilities

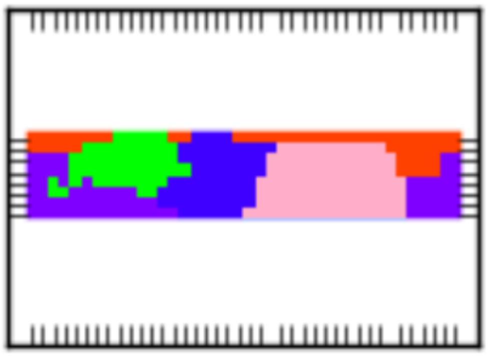
Utility methods available in ColorSims [14] and ColorSims 2.0, allow users to save and load populations from disk and to plot and save visual representations of a population. However, visual representation of the agents' naming strategies are different in ColorSims and ColorSims 2.0 due to the higher-dimensional color space. Differences are shown in Figure 2.

3 Examples of Results and Findings Using ColorSims 2.0

Section 3 summarizes some results and findings that were obtained using the novel features of ColorSims 2.0 described above. Because the sequence of pairwise agent interactions is randomized for each simulation run, the final



(a)



(b)

Figure 2: Visual representation of a single agent's color lexicon assuming a one dimensional color space (Fig. 2a, left) or a two-dimensional color space (Fig. 2b, right). In Figure 2a, there are 20 chips which are represented on the x-axis and 5 available color words represented on the y-axis. The blue line represents the name which the agent assigns to a given color chip (note: only 4 of the 5 color names are currently being used by this agent). In Figure 2b, the array represents the two-dimensional set of stimuli, where each box represents a single stimulus chip. The color of each box represents the name which the agent assigns to that given color chip.

simulation solutions, though robust, are not identical from one simulation to the next. Therefore, for the random populations and for each language in a subset of WCS languages, we ran all of the following measures through numerous simulations to verify their accuracy across rounds. All measures were found to be consistent across rounds.

3.1 Random Populations

Simulations using the updated features of higher-dimensional hue space and utilizing the stability measure as a criterion for stopping the simulation were first run on a population of random agents. After running multiple simulations and varying parameters such as population size and number of color words in the agents' vocabulary, all simulations were able to reach a stable solution (Figure 3). We can definitively say that the final color naming systems of these simulations were stable solutions based on our definition as determined by the stability measure S_r (see *Section 2.3*).

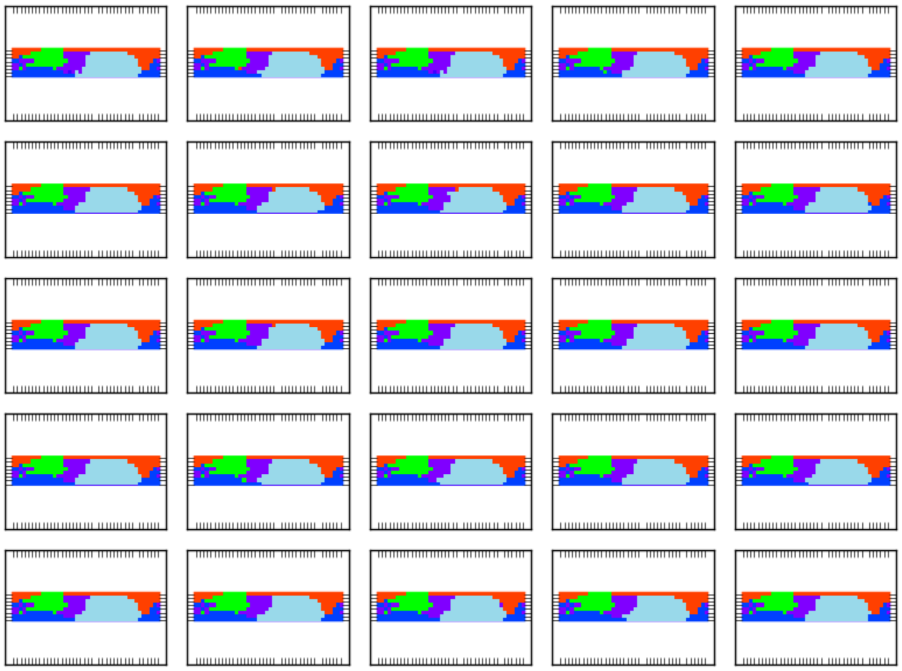


Figure 3: Final naming solution of a initial random population of 25 agents. Number of words in vocabulary = 5, k-sim = 63, network type: complete. Each cell outlined in black represents an agent in the population. The rectangle enclosed in each cell represents the 320 chromatic Munsell color chips used as the stimulus set. Each chip in the color array is colored according to the word that the agent is most likely to assign to that chip—the colors displayed in the figure are arbitrary representations of the color words in the agents' vocabulary. Simulation ran for 7.4 million rounds before reaching stable solution.

By mapping the population's global agreement as the simulation progressed, we found that the plot resembled that of a standard S-shaped learning curve, as in Figure 4. This indicates that agents were able to successfully learn a common, shared color naming system. It should be noted that the theoretical maximum value for A_r is 320, meaning that every agent is in complete agreement about what names to assign to all chips in the stimulus set. However, due to perceptual ambiguity that necessarily occurs on the boundaries between continuous color categories, the empirical values for A_r will tend towards 320 asymptotically.

Similarly, S_r approaches zero but will never remain at zero due to the

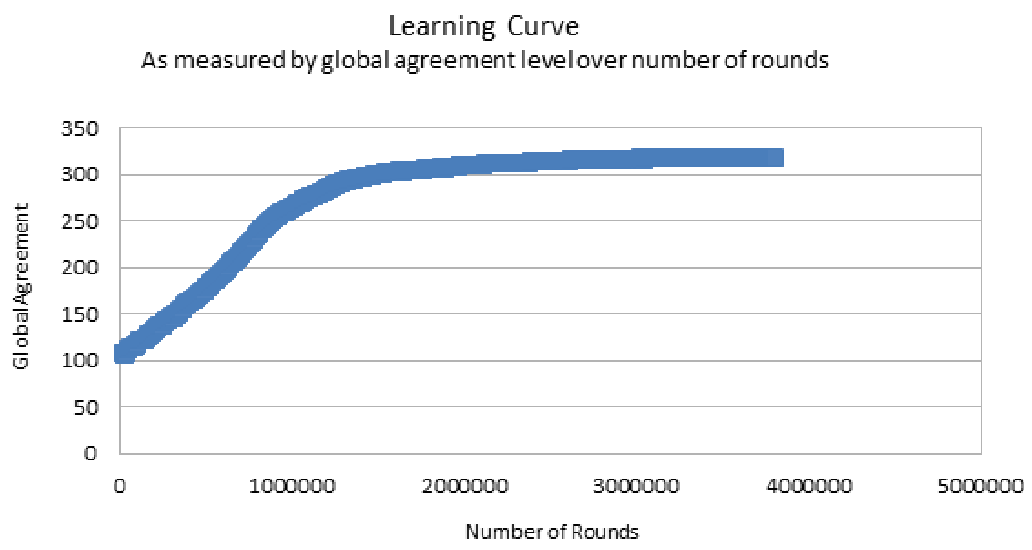


Figure 4: Plot of global agreement agreement measure (A_r) of a population of random agents across the number of rounds played. The shape of the curve closely resembles that of a standard learning curve.

boundary confusions (Figure 5). Since the S_r cannot stay at zero indefinitely, we defined a neighborhood around zero for which the solution is defined as stable, though not fixed. The default range mentioned in *Section 2.3*, (-0.00175, 0.00175), was determined empirically after running multiple simulations with varying parameters. It was found that most of these simulations had stability measures which converged into the range of ±0.00175. Though this value works for most simulations, it is not a universal value. Depending on the parameters given to the simulation, further empirical research must be done to establish a new value.

3.2 World Color Survey Language Populations

WCS simulations were run for all languages in the World Color Survey that were determined to have 3, 4, or 5 Basic Color Terms (BCTs) according to the Berlin and Kay definition of "basicness" [1]. Similar to the random population simulations, the WCS simulations also evolved to a stable solution and followed the S-shaped learning curve. The number of rounds required to reach stability was often shorter than the random population which corresponds

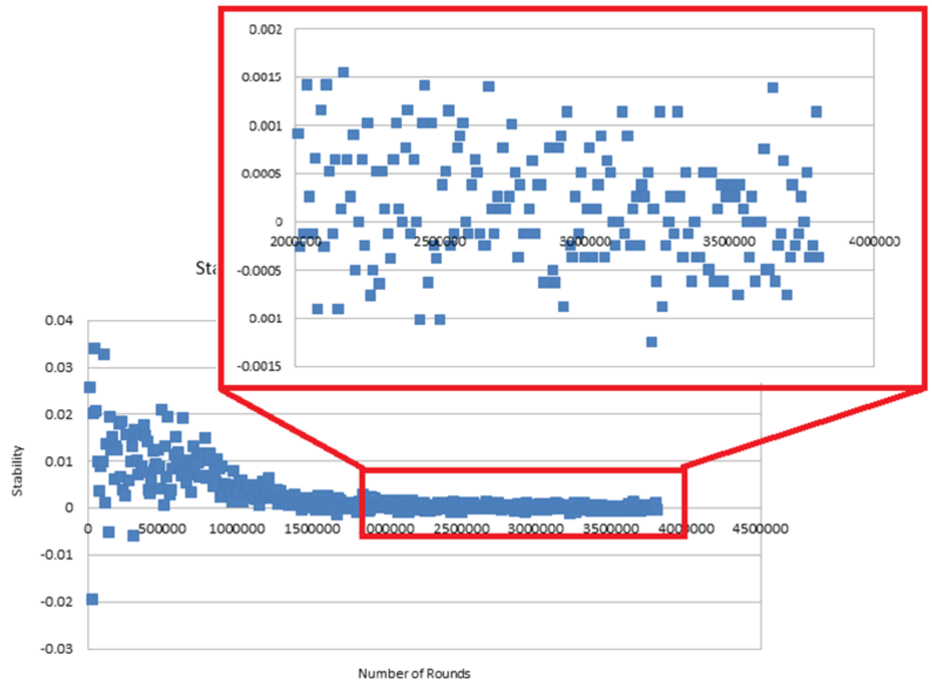
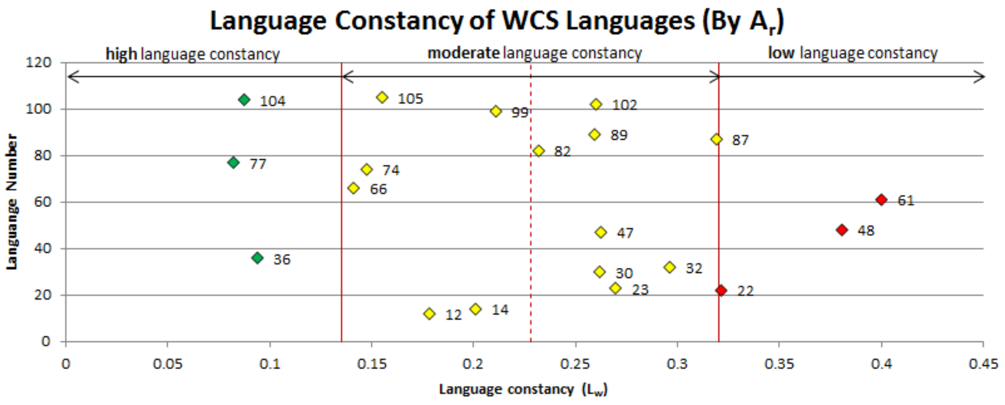


Figure 5: Plot of the stability measure (S_r) across numbers of rounds played. We define a "stable" solution to be one which falls within some "stability range" for many rounds. Due to the confusions that agents will have at the boundaries of two color words, the solution will be perpetually subject to minor fluctuations and changes. Thus, we say that a solution which has given a stability measure within some neighborhood of zero for enough rounds is "stable".

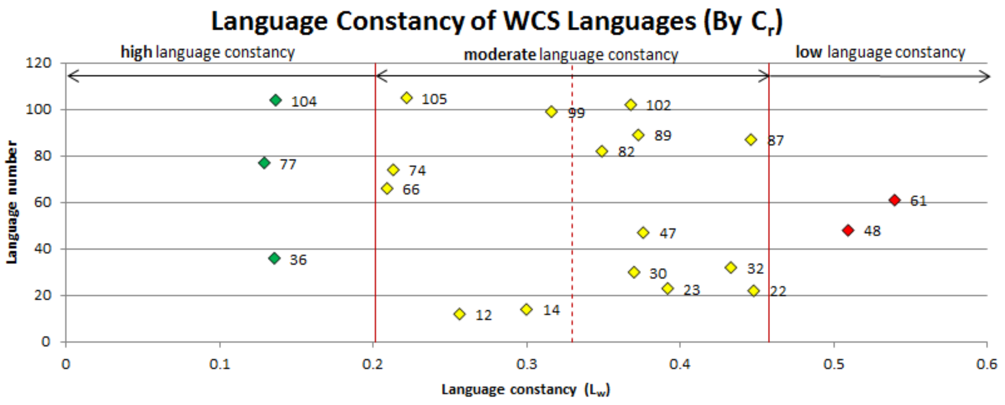
with our hypothesis: random populations would require more interactions to reach stability as opposed to WCS populations that have higher levels of agreements at the first round of the simulation.

Due to the diverse naming systems gathered from various linguistic societies in the World Color Survey, it was of interest to test the stability of the naming systems elicited at the time of the survey. Language constancy, L_w, was calculated for all languages in the 3, 4, 5 BCT set using two different measures of lexicon agreement—A_r for agreement by color chip and C_r for pairwise agreement among agents. Results are detailed in Table 1 and in Figures 6a and 6b.

The resulting classification of language constancy levels (high, moderate,



(a)



(b)

Figure 6: Plot of language constancy measure L_w for WCS languages with 3-5 BCTs calculated according to global agreement A_r (Fig. 6a, top) and according to pairwise agent correlation L_w (Fig. 6b, bottom). The numbers beside the data points refer to the number assigned to that language according to the WCS. The dotted red line in the middle of the plot represents the mean L_w value for this set of languages and the solid red lines represent one standard deviation from the mean. Languages with high language constancy have $L_w \in (0, \mu - \sigma)$, languages with moderate language constancy have $L_w \in [\mu - \sigma, \mu + \sigma]$, and languages with low language constancy have $L_w \in (\mu + \sigma, 1)$.

Table 1: Language Constancy Measure for WCS Languages (3-5 BCTs)

WCS Lang #	$\mathbf{L_w(byA_r)}$	$\mathbf{L_w(byC_r)}$
12	0.17825	0.25620
14	0.20088	0.29973
22	0.32138	0.44771
23	0.2695	0.39170
30	0.26175	0.36981
32	0.29613	0.43283
36	0.09388	0.13559
47	0.26238	0.37578
48	0.38063	0.50917
61	0.4	0.53946
66	0.141	0.20899
74	0.1475	0.21303
77	0.08211	0.12908
82	0.23188	0.34890
87	0.31913	0.44591
89	0.25926	0.37261
99	0.21088	0.316
102	0.26	0.36767
104	0.08738	0.13646
104	0.15501	0.22177

low) for each of the languages tested (i.e. those identified as having 3, 4, or 5 BCTs) is consistent whether using C_r or A_r to measure population lexicon agreement. Since the results are independent of the measure of agreement, we find that L_w is a robust measurement and, therefore, can calculate L_w using A_r exclusively without altering or skewing the results. The one exception to complete consistency in the classifications depicted in Figures 6a and 6b is for WCS Language 22. This is the only language in the set to change its constancy classification (moderate $\rightarrow$ low). However, this inconsistency is to be expected as Language 22 lies close to the cutoff boundary between moderate and low constancy. Natural variation due to the randomized agent pairing process in each simulation will cause Language 22 to oscillate within a small margin and across this boundary over repeated simulations.

Future work using WCS simulations will aim to contribute to the discussion regarding how best to define a languages number of BCTs, as presented in Fider et al. (2017) [2]. Whereas Berlin and Kay approached their analysis through a linguistic lense, Fider et al. employed numerical analysis which calculated the "basicness" of each term used in a society by looking only at individual participant responses. Though most of Fider et al.'s analysis corresponded with the original findings of Berlin and Kay, there were some languages with which the two analyses disagreed. By investigating such languages through a WCS simulation, we can bring new, simulation-based insight to this debate.

4 ColorSims 2.0 Availability

4.1 Operating system

The ColorSims 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.

4.2 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.

4.3 Dependencies

The core functionality depends heavily on the following external packages: Numpy [16], matplotlib [3] and networkx [12]. These packages must be properly installed and configured in your Python distribution. It is highly recommend the Anaconda Python distribution from Continuum Analytics (http://continuum.io/) which includes a number of preconfigured packages for scientific computing–including all three of the core dependencies for ColorSims.

4.4 Code

The code is available by request of the authors.

4.5 Instructions for Installation

STEP 1: Install Anaconda Python 2.7 Version at https://www.continuum.io/downloads.

Anaconda provides a preconfigured python installation containing libraries such as nympy and scipy, which are required for the ColorSims program, and other key libraries for scientific computing. It is highly suggested that users of the ColorSims program use Anaconda as opposed to installing and configuring python without it. Make sure to install 2.x version and NOT a 3.x version, as certain functions are not compatible with Python 3.x.

STEP 2: Install additional required libraries not installed in Anaconda Python by default.

Anaconda python has a command-line utility called conda that allows users to install/update/delete python packages. Most of the packages required by ColorSims are in the default Anaconda installation, but a couple of them need to be installed manually. From the command line enter this:

```
conda install progressbar
```

The library called moviepy is needed in order to build movies of the ColorSims simulations. Moviepy is not part of the Anaconda ecosystem and must therefore be installed with the standard python installer, called pip.

```
pip install moviepy
```

STEP 3: Unzip the colorsims repository received from the author. From the command line (inside the colorsims directory), users can run a demo simulation.

Launch the python command line:

```
python
```

From inside the python command line import the demo.py file:

```
import demo
```

**NOTE: The first time the file is imported it might take a bit longer than usual and install some additional python packages required by moviepy.*

Run the demo simulation:

```
demo.run_example_simulation()
```

Users should see some progress bars and, when it is done, a new directory called “demosim” which will contain the data for the simulation as well as another subfolder containing images of the agent population at intervals throughout the simulation.

There will also be a movie called “demosim.mp4” that animates the results of your simulation.

4.6 Code Modification

Finally, the code is based on object-oriented programming principles and it is relatively straight-forward to create new subclasses of the populations or to create new agent types that use different processes.

For example, there are already two different population network types: complete network and Moore-4 torus network. However, users can easily add new population subclasses that have different network structures. Users can also create new types of game dynamics that leverage the existing code base.

5 Acknowledgments

This work was in part funded by a University of California Pacific Rim Research Program award, 2010-2015 (Jameson, PI) and a grant for research from the National Science Foundation 2014-2018 (Award: SMA-1416907, K.A. Jameson, 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.

A Pairwise Agent Comparison measure

The pairwise agent comparison measure (C_r) is defined to be an alternative method for evaluating the level of agreement of a population's color categorization at a given time. Where as the global agreement measure (A_r) aggregates agreement across the set of color chips (see *Section 2.3*), pairwise agent comparison aggregates agreement across all unique pairs of agents. C_r is formally defined as follows:

$$C_r = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{k=1}^{320} [W_{ik} = W_{jk}]}{\frac{(N-1)N}{2}} \tag{5}$$

where i and j are different agents from the population, N is the population size, k is the chip number, r is the number of rounds, W_{ik} is the word that agent i assigned to color chip k, the number of possible pairs of agents is given by $\frac{(N-1)N}{2}$ (see Figure 7), and $[W_{ik} = W_{jk}] = \begin{cases} 1 & \text{if } W_{ik} = W_{jk} \\ 0 & \text{if } W_{ik} \neq W_{jk} \end{cases}$.

```
1,2
1,3   2,3
1,4   2,4   3,4
1,5   2,5   3,5   4,5
.     .     .     .     .
.     .     .     .     .     .
.     .     .     .     .     .     .
1,N   2,N   .     .     .     .     .     N-1,N
```

Figure 7: All possible pairs of agents used in calculating pairwise agent comparison (C_r).

Language constancy can then be calculated by pairwise agent comparison C_r as follows:

$$L_w = \frac{|C_0 - C_{\bar{r}}|}{320} \tag{6}$$

where C_0 is the pairwise agent comparison measure between unique dyads of the population at the beginning of the simulation and $C_{\bar{r}}$ is the measure at the end of the simulation.

References

[1] Berlin, B., Kay, P. *Basic color terms: Their university and evolution.* California UP, 1969

[2] Fider, N., Narens, L., Jameson, K.A., Komarova, N.L. *Quantitative approach for defining basic color terms and category best exemplars.* JOSA A, 34(8):1285–1300, 2017.

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

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

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

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

[7] Jameson K.A., Komarova N.L., Tauber S., Narens L. *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.

[8] Komarova N.L., Jameson K.A., Narens L. *Evolutionary models of color categorization based on discrimination.* Journal of Mathematical Psychology, 51(6):359–382, 2007.

[9] Komarova N.L., Jameson K.A. *Population heterogeneity and color stimulus heterogeneity in agent-based color categorization.* Journal of Theoretical Biology, 253(4):680–700, 2008.

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

[11] Regier, T., Kay, P., Cook, R. *Focal colors are universal after all.* Proceedings of the National Academy of Sciences of the United States of America, 102(23), 8386–8391, 2005.

[12] Schult D. A., Swart PJ. *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.

[13] Schwiegerling, J. *Field guide to visual and ophthalmic optics.* Spie, 2004.

[14] Tauber S. *ColorSims: A python package for evolving linguistic color naming conventions within a population of simulated agents.* Technical Report Series #MBS15-01. Institute for Mathematical Behavioral Sciences, University of California, Irvine, Irvine, CA, USA, 2015.

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

[16] Van DerWalt S., Colbert C. S., Varoquaux G. *The numpy array: a structure for efficient numerical computation.* Computing in Science & Engineering, 13(2):22–30, 2011.