APPENDIX A: MODEL

This provides a walkthrough behind the mathematical reasoning of the Kodner & Cerezo Falco (2018) model summarized in the main text.

Representing the Network

The network representation is required for calculating the probability by which every individual j interacts with any individual i to be calculated. How to interpret the edges in the network is a modeling choice left to the researcher, affording a great deal of flexibility: they could primarily represent social distance or judgment (Blythe & Croft, 2012; Fagyal et al., 2010) or physical geographic distance. Edges may be directed or undirected and may be weighted evenly between nodes or allowed to vary, which may signify relative social standing, for example (Blythe & Croft, 2012, section 3.3). Additionally, the whole network can be varied over time to indicate changes in the dynamics of the communities such as the formations of new connections, births, deaths, migrations, changing social opinions, and so on.

For the computer's purposes, the network is represented as an *adjacency matrix*, essentially a table that indicates which nodes are connected in which directions and with what weights. In a population of n individuals, it is an $n \times n$ matrix $\mathbf{A}$ where each element a_{ij} is the weight of the connection from individual j to individual i. All the entries in the matrix must lie between 0 and 1, and every column in the matrix needs to sum to 1 so that weights can be interpreted as probabilities. The matrix represents a network with only undirected edges if every entry a_{ij} equals a_{ji}, and it can capture a perfectly-mixed population when each entry equals $1/n$. Figure A1 provides a very simple example adjacency matrix and the network graph that it represents.

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

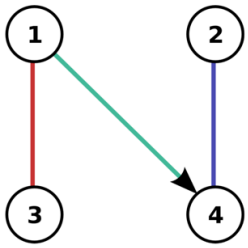

FIGURE A1. Simple adjacency matrix (not normalized) and corresponding network graph with two undirected (bidirectional) and one directed edge. Adjacency matrix values are color coded to math their corresponding edges.

Propagation in the Network

Our framework conceptually allows individuals to "travel" *k* steps along the graph structures to interact with individuals farther away from them, at every step deciding whether to stop and interact or to keep going. For present purposes, *k* is modeled according to a geometric distribution[1], and combining this with **A** yields the matrix defined by Equation A1 which gives the probability at which every individual agent would interact with any other individual. The α parameter from the geometric distribution ranges between 0 and 1 and dictates how quickly the probability of taking another step falls off. A large α means that agents travel less and a small α means that they travel more, so this is conceptually like defining how many grid spaces each agent would move per time step in a swarm framework. **I** is an $n \times n$ *identity matrix*, which is just a square matrix of zeros with ones down the diagonal from top left to bottom right.

EQUATION A1.

$$\alpha(\mathbf{I} - (1 - \alpha)\mathbf{A})^{-1}$$

An individual's linguistic environment is defined by whom that individual interacts with along with which linguistic variants those agents express, so we need a representation for the variants. Each individual is conceived of as having *g* linguistic variants to choose from, and each of the *n* individuals in the network has some distribution of variants. These are collected into an $n \times g$ matrix **G**. Each value in this matrix is between 0 and 1, and each row sums to 1 so that

each individual is represented with a probability distribution over variants. The number of variants available, whether they are continuous or discrete, as well as whether individuals must apply one categorically or can express a distribution over them are all dependent on the specific problem. For a continuous variable in one dimension such as what might be appropriate for representing the mean production position of a vowel, g equals 2, and each row is interpreted as a point in a simple vowel space. Figure A2 visualizes this with a small **G** matrix. In a situation where a continuous interpretation is appropriate, such as mean position of vowel productions along one dimension, the figure indicates one speak with a mean 37% through the possible range, another 82%, and a third 63%.

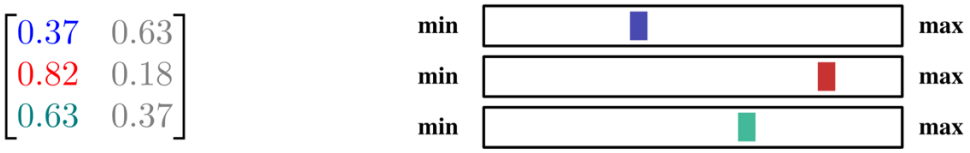

FIGURE A2. 3-speaker **G** matrix demonstrating one-dimensional continuous variables.

Finally, Equation A1 is combined with **G** to yield an *environment function* $\mathcal{E}_n(\mathbf{G}_t, \mathbf{A}_t)$ shown in Equation A2. This outputs one more matrix, $\mathbf{E}_{t+1}$, in the shape of $\mathbf{G}^\top$ that specifies the proportion of each linguistic variant in each individual's environment.

EQUATION A2.

$$\mathcal{E}_n(\mathbf{G}_t, \mathbf{A}_t) = \mathbf{E}_{t+1} = \mathbf{G}_t^\top \alpha(\mathbf{I} - (1-\alpha)\mathbf{A})^{-1}$$

Learning in the Network

The environment function $\mathcal{E}_n$ describes what mix of variants the individuals are exposed to, but it does not describe how they respond to them. That is the task of what we call, without loss of generality, the *learning algorithm*, a function $\mathcal{A}(\mathbf{E}_{t+1}) = \mathbf{G}_{t+1}$ which takes each individual's environment as specified by a column in $\mathbf{E}_{t+1}$ and turns it into a description of each individual's

language as specified by a row in $\mathbf{G}_{t+1}$. Since $\mathbf{G}_{t+1}$ is defined as $\mathcal{A}(\mathbf{E}_{t+1})$ and is an argument to $\mathbf{E}_n$, this captures the back-and-forth that is the two-step cycle of language change. The previous equation can now be rewritten in the form of a dynamical system as in Equation A3. This is the core of our framework that allows us to directly calculate what would have happened if we had simulated each agent individually as in Algorithm 1. The learning algorithm $\mathcal{A}$ conceivably describes any function that takes probabilities as inputs and provides probabilities as outputs, so the specific choice of $\mathcal{A}$ depends on theoretical concerns and the problem at hand. It does not automatically impose a difference between child learners and adults.

EQUATION A3.

$$\mathcal{E}_n(\mathbf{G}_t, \mathbf{A}_t) = \mathbf{E}_{t+1} = \mathcal{A}(\mathbf{E}_t)^\top \alpha(\mathbf{I} - (1-\alpha)\mathbf{A})^{-1}$$

Example Adjacency Matrix

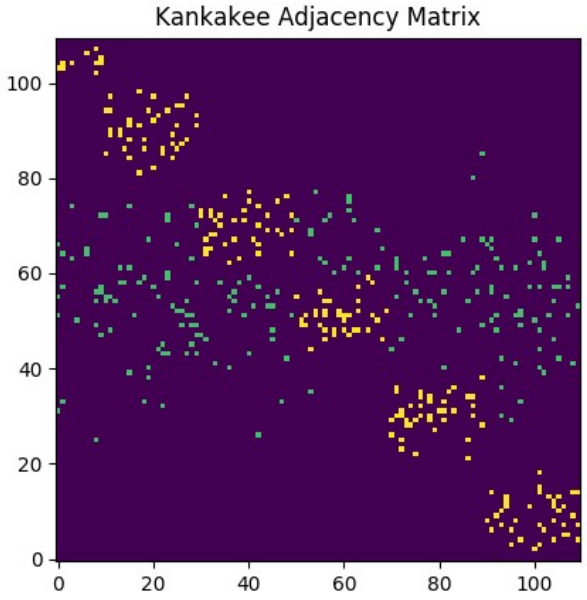

FIGURE A3. Representation of the Kankakee, IL section of the adjacency matrix sorted by cluster so that they fall along the diagonal. Location (i, j) indicates whether there is a connection between node j and node i. This should not be read as a geographic layout of the city. Intra-cluster connections are yellow and inter-cluster connections are green. Weight is not indicated.

The band of green inter-cluster is the result of sampling indices from a clipped normal distribution. Individuals represented by rows in that band have substantially more connections with other clusters than those outside of it. The clusters themselves are generated the same way (with standard deviations in the code). You can then imagine the city as densely connected groups of individuals with some clusters connected to virtually all other clusters (they are well-integrated, well-socially connected, "popular" even, groups in the community), while some contain members that tend not to have connections to many other groups in the community.

When connections are being laid down, a pair (i, j) is selected and the value at that location is incremented. If the same (i, j) pair is selected again, that value is incremented again. At the end everything is normalized so that it can be treated as probabilities.

Connections between cities then proceed similarly. For any two cities that are connected on the map, (i, j) pairs are selected and incremented.

APPENDIX B: SIMULATION

This section contains additional details on modeling parameters.

- Learning algorithm $\mathcal{A}(\mathbf{E}_{t+1}) = \mathbf{G}_{t+1}$ maps linguistic environments to grammars.
- α indicates the fluidity of movement in the network. Higher α values yield faster simulations. The parameter was set such that iterations can be interpreted roughly as years in each simulation.
- The proportion of nodes that will be updated at each iteration is set to 0.1 for every simulation.
- m[*source*, *dest*] indicates rate of migration between two regions. Migration is implemented as replacement of the specified fraction of nodes' *g* in *dest* with node values *m*. 100% of migrating Chicago nodes exhibit NCF, and 0% of Midlands nodes exhibit NCF.
- σ (standard deviation) determines how centralized clusters are in the network. Higher σ values yield less clustered networks.
- *k* values specify the number of edges to be laid down in the network as a function of adjacency matrix size and therefore the rate of interaction (diffusion) among nodes.

Simulation 1

Learning algorithm:
Neutral change ($\mathbf{G}_{t+1} = \mathbf{E}_{t+1}$)

Production:
Individuals make continuous productions (*g* is not categorical)

Interaction:
$\alpha = 0.2$

Network generation:
$\sigma = 0.2$
individual $k = 0.1$
intra-city cluster $k = 0.02$

cross-city cluster *k*:
stage 1 = 0.1
stage 2 = 0.01
stage 3 = 0.1
Migration:
m[Chicago, On-Route]: 0.0
m[Midlands, On-Route]: 0.0
m[Midlands, Off-Route]: 0.0
m[On-Route, Off-Route]: 0.0

Stage Durations:
stage 1 = 5
stage 2 = 15
stage 3 = 45

Simulation 2

Geographic Model is identical to Simulation 1 except:

Migration:
m[Chicago, On-Route]:
Stage 1: 0.0
Stage 2: 0.02
Stage 3: 0.0
m[Midlands, On-Route]:
Stage 1: 0.1
Stage 2: 0.0
Stage 3: 0.1
m[Midlands, Off-Route]:
Stage 1: 0.1
Stage 2: 0.01
Stage 3: 0.1
m[On-Route, Off-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.0

Schematic Model

Interaction:
$\alpha = 0.45$

Network generation:
$\sigma = 0.2$
individual $k = 5$

intra-city cluster k = (none)
cross-city cluster k = 0.5

Migration:
m[Chicago, On-Route]:
Stage 1: 0.0
Stage 2: 0.02
Stage 3: 0.0
m[Midlands, On-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.05
m[Midlands, Off-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.0
m[On-Route, Off-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.0

Stage Durations:
stage 1 = 5
stage 2 = 20
stage 3 = 75

Simulation 3

Learning algorithm:
Advantaged change (see Niyogi & Berwick [1996, 1997] for derivation): Acquire NCF proportionally to its expression in the environment but biased towards more NCF. This bias is accomplished as follows with a = 0.3, b = 0.5 (roughly, a moderate bias).

Define transition matrices $\mathbf{T}_0$ (non-NCF to NCF) and $\mathbf{T}_1$ (NCF to non-NCF). If $a < b$, acquisition is advantaged in favor of NCF.
$\mathbf{T}_0$ (non-NCF to NCF) = [1, 0; 1-a, a]
$\mathbf{T}_1$ (non-NCF to NCF) = [b, 1-b; 0, 1]

Given $\mathbf{G}_{t+1}$ and the set of $\mathbf{T}$, it is possible to calculate G_t+1 as the behavior of the learners in the limit. Define, $\mathbf{G}_{t+1}$ = [g_0, g_1], then,

g_i = dominant eigenvector of $\mathbf{E}_{t+1;0,i}\mathbf{T}_0 + \mathbf{E}_{t+1;1,i}\mathbf{T}_1$.

All other parameters are identical to Simulation 2's Geographic Model except for the following:

Matrix cluster generation:

cross-city cluster *k*:
stage 1 = 0.01 or 0.00001
stage 2 = 0.01 or 0.00001
stage 3 = 0.1
stage 4 = 0.1

Migration:
m[Chicago, On-Route]:
Stage 4: 0.0
m[Midlands, On-Route]:
Stage 4: 0.1
m[Midlands, Off-Route]:
Stage 4: 0.1
m[On-Route, Off-Route]:
Stage 4: 0.0

Stage Durations:
stage 1 = 5
stage 2 = 15
stage 3 = 20
stage 4 = 25

Simulation 4

Parameters are identical to Simulation 1 except for the following

Learning algorithm:
Neutral change ($\mathbf{G}_{t+1} = \mathbf{E}_{t+1}$)
Nodes updated in the last three iterations not my migration are considered children.

Network generation:
cross-city cluster *k*:
stage 4 = 0.03
stage 5 = 0.06

Migration:
m[Chicago, On-Route]:
Stage 1: 0.0
Stage 2: 0.02
Stage 3: 0.0
Stage 4: 0.0
Stage 5: 0.0
m[Midlands, On-Route]:
Stage 1: 0.1
Stage 2: 0.0
Stage 3: 0.1

Stage 4: 0.1
Stage 5: 0.1
m[Midlands, Off-Route]:
Stage 1: 0.1
Stage 2: 0.1
Stage 3: 0.1
Stage 4: 0.0
Stage 3: 0.1
m[On-Route, Off-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.0
Stage 4: 0.06
Stage 5: 0.0
Migration form On-Route to Off-Route imports NCF at a rate of 20%.

Stage Durations:
stage 1 = 5
stage 2 = 15
stage 3 = 10 or 0
stage 4 = 15 or 25
stage 4 = 20

Simulation 5

All parameters are identical to Simulation 4 except for the following:

Learning:
Threshold Learning (see Yang [2009] for derivation): If the proportion of NCF in an individual's environment is greater than the pre-specified threshold (0.3, 0.5, 0.8), then categorically acquire NCF, otherwise categorically acquire non-NCF.

Production:
Individuals make categorical productions (*g* is categorical)

Network generation:
cross-city cluster *k*:
stage 1 = 0.01
stage 2 = 0.001
stage 3 = 0.01
stage 4 = 0.03
stage 5 = 0.06
Migration:
m[Chicago, On-Route]:
Stage 1: 0.0

Stage 2: 0.02
Stage 3: 0.0
Stage 4: 0.0
Stage 5: 0.0

m[Midlands, On-Route]:
Stage 1: 0.1
Stage 2: 0.0
Stage 3: 0.0
Stage 4: 0.0
Stage 5: 0.0

m[Midlands, Off-Route]:
Stage 1: 0.1
Stage 2: 0.0
Stage 3: 0.0
Stage 4: 0.0
Stage 3: 0.0

m[On-Route, Off-Route]:
Stage 1: 0.0
Stage 2: 0.0
Stage 3: 0.0
Stage 4: 0.06
Stage 5: 0.0

Appendix Notes

1. A variant where k decays by a Poisson distribution yields qualitatively similar result.