SIMULATING THE EVOLUTION OF CONSONANT INVENTORIES

by

JAMES SCOTT MACKIE

B.A., University of Ottawa, 2006
M.A., University of Ottawa, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Linguistics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

January, 2017

©James Scott Mackie 2017
Abstract

A major question in phonology concerns the role of historical changes in shaping the typology of languages. This dissertation explores the effect of sound change on consonant inventories.

Historical reconstruction is mainly done by comparing cognate words across languages, making it difficult to track how inventories change specifically. Additionally, few languages have historical written records that can be directly examined. For this dissertation, the main research tool is computer simulation, using bespoke software called PyILM, which is based on the Iterated Learning Model (Kirby 2011, Smith et al. 2003). This allows for the simulation of sound change from arbitrary starting points, controlling for a multitude of variables.

PyILM is an agent-based model, where a ‘speaking’ agent transmits a set of words to a ‘listening’ agent. The speaking agent is then removed, the learner becomes the speaker, and a new learner is introduced. The cycle repeats any number of times, roughly simulating the transmission of language over many generations.

Sound change in a simulation is due to channel bias (Moreton 2008), the result of which is that agents occasionally misinterpret some aspect of speech, and internalize sound categories that differ from the previous generation (Ohala 1981, Blevins 2004). Three typological generalizations are examined, none of which have previously been studied from an evolutionary perspective:

1. The total number of consonants in a language. This is shown to be related to syllable structure, such that languages with simple syllables develop smaller inventories than languages with complex syllables. This mirrors a positive correlation between inventory size and syllable structure in natural languages, as reported by Maddieson (2007).

2. The correlation reported by Lindblom and Maddieson (1988) between the size of an inventory and the complexity of its segments. This effect emerges in simulations when context-free changes are introduced, since these changes produce similar outcomes in inventories of all sizes.

3. Feature economy (Clements 2003), which refers to the way that consonants within a language tend to make use of a minimal number of distinctive features. Economy emerges over time when sound changes take scope over classes of sounds, rather than targeting individual sounds.
Preface

This dissertation is the original work of the author. I wrote all of the computer code for Py-ILM, including the accompanying GUI, from the ground-up using Python 3.4. The Feature Economist algorithm used for the results in Chapter 5 was originally designed in collaboration with Jeff Mielke. It has previously been used for the research in Mackie and Mielke (2011), and it is included with the software P-base (Mielke 2008). The implementation used for this dissertation is one that I wrote myself.
Table of Contents

Abstract ................................................................. ii
Preface ................................................................. iii
Table of Contents ..................................................... iv
List of Tables ........................................................ ix
List of Figures ........................................................ x
List of Algorithms .................................................... xii
Acknowledgments ...................................................... xiii

1 Consonant inventories and sound change ......................... 1
  1.1 Introduction ..................................................... 1
  1.2 Iterated learning models ....................................... 3
  1.3 Sound change models .......................................... 8
      1.3.1 Summary .................................................. 17
  1.4 An ILM for phonology .......................................... 17
      1.4.1 Overview ................................................ 17
      1.4.2 One turn of a PyILM simulation ......................... 18
          1.4.2.1 Production ......................................... 18
          1.4.2.2 Misperception ..................................... 18
          1.4.2.3 Learning ........................................... 20
  1.4.3 Some notes on design ..................................... 20
      1.4.3.1 Social factors ....................................... 20
      1.4.3.2 Single-agent transmission change ................. 21
      1.4.3.3 Discrete learning period ......................... 22
      1.4.3.4 No teleology ........................................ 22
      1.4.3.5 Phonemes and allophones ......................... 23
  1.4.4 Expected outcomes and inventory structure ............... 24
  1.5 Summary ........................................................ 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.9</td>
<td>Agents</td>
<td>43</td>
</tr>
<tr>
<td>2.2.9.1</td>
<td>lexicon</td>
<td>44</td>
</tr>
<tr>
<td>2.2.9.2</td>
<td>inventory</td>
<td>44</td>
</tr>
<tr>
<td>2.2.9.3</td>
<td>feature_space</td>
<td>44</td>
</tr>
<tr>
<td>2.2.9.4</td>
<td>distributions</td>
<td>44</td>
</tr>
<tr>
<td>2.2.10</td>
<td>Misperception</td>
<td>45</td>
</tr>
<tr>
<td>2.2.10.1</td>
<td>name</td>
<td>46</td>
</tr>
<tr>
<td>2.2.10.2</td>
<td>target</td>
<td>46</td>
</tr>
<tr>
<td>2.2.10.3</td>
<td>feature</td>
<td>46</td>
</tr>
<tr>
<td>2.2.10.4</td>
<td>salience</td>
<td>46</td>
</tr>
<tr>
<td>2.2.10.5</td>
<td>env</td>
<td>46</td>
</tr>
<tr>
<td>2.2.10.6</td>
<td>p</td>
<td>47</td>
</tr>
<tr>
<td>2.2.10.7</td>
<td>How misperception happens</td>
<td>47</td>
</tr>
<tr>
<td>2.2.10.8</td>
<td>A note on misperception definitions</td>
<td>48</td>
</tr>
<tr>
<td>2.3</td>
<td>Algorithms</td>
<td>48</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Learning algorithm</td>
<td>49</td>
</tr>
<tr>
<td>2.3.1.1</td>
<td>Parsing a Word</td>
<td>49</td>
</tr>
<tr>
<td>2.3.1.2</td>
<td>Creating new segment categories</td>
<td>52</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Updates</td>
<td>53</td>
</tr>
<tr>
<td>2.3.2.1</td>
<td>The lexicon</td>
<td>53</td>
</tr>
<tr>
<td>2.3.2.2</td>
<td>The inventory</td>
<td>53</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Determining phonological feature values</td>
<td>53</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Production algorithm</td>
<td>55</td>
</tr>
<tr>
<td>2.3.4.1</td>
<td>Initialization</td>
<td>55</td>
</tr>
<tr>
<td>2.3.4.2</td>
<td>Step 1: Word selection</td>
<td>56</td>
</tr>
<tr>
<td>2.3.4.3</td>
<td>Step 2: Transforming Segments into Sounds</td>
<td>56</td>
</tr>
<tr>
<td>2.3.5</td>
<td>Invention algorithm</td>
<td>57</td>
</tr>
<tr>
<td>2.4</td>
<td>Using PyILM</td>
<td>58</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Obtaining PyILM</td>
<td>58</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Configuration files</td>
<td>58</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Running a simulation</td>
<td>60</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Viewing results</td>
<td>61</td>
</tr>
<tr>
<td>2.5</td>
<td>Other notes</td>
<td>62</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Limitations</td>
<td>62</td>
</tr>
<tr>
<td>2.5.1.1</td>
<td>No social contact</td>
<td>62</td>
</tr>
<tr>
<td>2.5.1.2</td>
<td>No deletion or epenthesis</td>
<td>63</td>
</tr>
<tr>
<td>2.5.1.3</td>
<td>No morphology or syntax</td>
<td>64</td>
</tr>
<tr>
<td>2.5.1.4</td>
<td>No long distance changes</td>
<td>64</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Running time</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>Sample simulations</td>
<td>66</td>
</tr>
</tbody>
</table>
## 4 Natural Language Consonant Inventories

4.1 Inventory size .................................................. 84
   4.1.1 Overview .................................................. 84
   4.1.2 Population size ............................................ 90
   4.1.3 Hypothesis #1: Phonotactics and inventory size ...... 98
4.2 Inventory contents .............................................. 100
   4.2.1 Overview .................................................. 100
   4.2.2 Hypothesis #2 - Common Consonants ................. 107
4.3 Inventory organization ................................ .......... 107
   4.3.1 Overview .................................................. 107
   4.3.2 Feature economy ......................................... 109
      4.3.2.1 Measuring feature economy ....................... 112
   4.3.3 Cross-linguistic tendencies .............................. 115
   4.3.4 Explaining economy ..................................... 119
      4.3.4.1 A computational model .......................... 120
      4.3.4.2 Whistle experiments ............................. 122
   4.3.5 Hypothesis #3 - Sound change and feature economy .. 125
4.4 Summary ....................................................... 126

## 5 Simulating Inventory Evolution .................................. 128

5.1 Introduction .................................................. 128
5.2 Inventory size ................................................ 128
   5.2.1 Simulation results ...................................... 131
5.3 Common Consonants ............................................ 132
   5.3.1 Misperception vs. bias ................................ 133
   5.3.2 Simulation results ...................................... 137
5.4 Feature economy ............................................... 140
   5.4.1 How economy can change over time .................... 141
   5.4.2 An illustrative example ................................ 146
   5.4.3 Segment-specific misperceptions vs. class-level misperceptions ........................................ 148
   5.4.4 Calculating feature economy ........................... 150
   5.4.5 Simulation results ...................................... 152
6 Conclusion .......................................................... 156
Bibliography ........................................................ 158
List of Tables

3.1 Configuration for Simulation 1 ........................................ 67
3.2 Comparison of inventories in Simulation 1 .......................... 68
3.3 Comparison of inventories in Simulation 2 .......................... 72
3.4 Comparisons of several generations in Simulation 3 .............. 75
3.5 Configuration for Simulation 4 ........................................ 77
3.6 Comparison of several generation in Simulation 4 .................. 80
3.7 Configuration for Simulation 5 ........................................ 81
3.8 Comparison of several generations in Simulation 5 .............. 82

4.1 Co-occurrence of V and Z in UPSID (from Clements (2003, p. 303)) .... 111
4.2 The inventory of West Greenlandic ................................... 117
4.3 Feature economy effects in Pater and Staub (2013) .................. 121

5.1 Configuration for testing phonotactic effects on inventory size .... 129
5.2 Configuration for simulations comparing simple misperceptions and biases . 136
5.3 Example of individual inventories in a simulation with misperception and bias, starting from only voiceless stops .................. 138
5.4 Misperceptions and biases for testing Hypothesis #2 ............... 139
5.5 Results of two-way ANOVA with inventory size and misperception type as predictors and economy score as dependent variable. .... 154
# List of Figures

1.1 Model of sound change through listener misperception, Ohala (1981, p. 182) .................. 10
1.2 Emergent stops, from Ohala (1997) .............................................................. 11

2.1 The objects of PyILM ................................................................. 29
2.2 The transmission of a phonological segment .................................................... 30
2.3 Sample feature file ................................................................................. 34
2.4 Example configuration file ............................................................................ 59
2.5 Screen shot of PyILM Visualizer ............................................................... 62

3.1 Change in inventory size for Simulation 1 ...................................................... 70
3.2 Results for various values of `minimum_activation_level` ...................................... 73
3.3 Varying misperception salience across three different values for
`minimum_activation_level`. Misperception salience is shown in the
legend. Simulation (a) uses a value of 0.2, Simulation (b) uses a value of 0.5
and Simulation (c) uses a value of 1.0 ......................................................... 74
3.4 Change in inventory size for Simulation 3 ...................................................... 76
3.5 Change in total inventory size with five different random seeds ......................... 79

4.1 The inventories of Palauan, from Morén-Duolljá (2005) ........................................ 85
4.2 Summary of the distribution of velar stops in Palauan, with data from Morén-Duolljá (2005)) ................................................................. 86
4.3 Phonemic inventory of Central Rotokas, based on Firehow and Firehow (1969) 88
4.4 The inventory of !Xóó, based on Traill (1985) .................................................. 89
4.5 Correlations between speaker population size (individual languages) and
inventory size, from Hay and Bauer (2007). ................................................... 91
4.6 Correlations between speaker population size (language families) and
inventory size, from Hay and Bauer (2007) ................................................... 92
4.7 Relationship between population size (log scale) and inventory size for several
language families, from Donohue and Nichols (2011) ..................................... 94
4.8 Population size and inventory size, from Wichmann, Rama and Holman (2011) 95
4.9 Predicted magnitude of the effect of population size on inventory size, from
Moran et al. (2012, p. 18) ........................................................................... 96
<table>
<thead>
<tr>
<th>Page</th>
<th>Section Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>4.10 IPA chart warped to show consonant frequency in P-base (Mielke 2008)</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>4.11 Consonant inventory size and number of superset inventories in P-base</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>4.12 Consonant inventory size in P-base and number of unique consonants</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>4.13 Segment complexity plotted against inventory size for the inventories of P-base</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>4.14 Consonant inventories from P-base with “reversed” segment complexity</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>4.15 Consonant inventories of Noon and Tamazight</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>4.16 Randomly generated consonant inventory</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>4.17 Three sound systems differing in symmetry and economy, from Clements (2003, p. 292)</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>4.18 Inventories of Hawaiian, French and Nepali (from Clements (2003, p. 288))</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>4.19 Ranges of feature economy scores in the inventories of P-base (Mielke 2008)</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>4.20 Feature economy scores of natural languages and randomly generated inventories</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>4.21 Example of whistle recombinations from Verhoef and de Boer (2011, p. 2)</td>
<td></td>
</tr>
</tbody>
</table>

| 131  | 5.1 Average inventory size for 50 simulations over 50 generations, across 3 different phonotactic conditions. |
| 134  | 5.2 State diagram for word-final obstruents in a simulation with final devoicing      |
| 138  | 5.3 Change in inventory size for two simulations, one starting with voiceless stops, one with voiced stops |
| 140  | 5.4 Biased and non-biased sounds in the final simulated inventories.                  |
| 143  | 5.5 Change in economy score for a hypothetical language                              |
| 145  | 5.6 Range of possible Simple Ratio scores                                           |
| 145  | 5.7 Range of possible Frugality scores                                              |
| 147  | 5.8 Change in feature economy for a simple simulation                               |
| 153  | 5.9 Change in average feature economy for simulations run with class-level changes  |
| 153  | 5.10 Change in average feature economy for simulations run with segment-specific changes |
# List of Algorithms

1.1 Generalized Iterated Learning Model ................................. 5  
1.2 Generalized Iterated Learning Model for phonology .............. 18  
2.1 Main simulation loop ................................................. 27  
2.2 Misperception function .............................................. 47  
2.3 Learning algorithm .................................................. 50  
2.4 Activation function .................................................. 51  
2.5 K-means algorithm .................................................. 54  
2.6 Distribution estimation .............................................. 56  
2.7 Production algorithm ............................................... 57  
2.8 Invention algorithm ............................................... 57
Acknowledgments

This project has been a long time in the making, and it could not have been completed without the help of many people. First and foremost is my family. My partner Cameron and our daughter Autumn have shown incredible patience, love, and understanding over the years, and I could not have finished this without their support. My parents, Margaret and Craig, and my brothers, Simon and Alan, have been extremely supportive and encouraging, even if they don’t always understand what I’m up to. I love you all.

I’d also like to thank my committee, Gunnar, Molly, and Alex, who have all been extremely helpful over the course of this project. I’m grateful for all that they have done for me. Gunnar was my first contact at UBC when I applied to be a PhD student, and I’m extremely happy that he was able to direct my dissertation as well.

There are many other individuals who made my graduate school experience special, and who deserve a mention too. In no particular order, I’d like to thank...

Jeff Mielke and Ana Arregui, at the University of Ottawa, for encouraging me to pursue further studies in linguistics.

Xiahou Dun, Kraus, a foolish pianist, Robo Kitty, Cingulate, Mortley, !amicable and others on the SA forums who have really helped me refine my ideas about language and linguistics.

The PCT team: Kathleen Hall, Blake Allen, Michael Fry, and Michael McAuliffe, who taught me a great deal about programming, and helped me see how much fun it is.

Johnathon Jones, who has given me amazing opportunities to apply linguistics outside of a university setting.

Everyone I’ve ever TAed for (Susannah Kirby, Lisa Matthewson, Brian Gick, Douglas Pulleyblank, Henry Davis, and especially Strang Burton). Teaching has been one of the most meaningful and exciting parts of my PhD, and I’ve enjoyed every class with every one of you.
Chapter 1

Consonant inventories and sound change

1.1 Introduction

Each human language uses only a finite subset of all possible consonants and vowels, and this collection of sounds is known as the inventory of the language. This dissertation is a study of consonant inventories. I investigate three different aspects of consonant inventories, and how they change over time.

The first is the total size of an inventory. I propose that a main factor that influences inventory size is the phonotactics (syllable structure) of a language. Languages with more restrictive phonotactics (e.g. only CV syllables, and hence no word-internal consonant clusters, nor any final consonants) will tend to develop small inventories over time, while languages with more permissive phonotactics (e.g. maximally CCVCC syllables, and hence the possibility of word-internal consonant clusters as well as final consonants) will develop larger inventories. This is supported by a correlation reported in Maddieson (2007), which shows that syllable structure complexity is positively related to inventory size in the inventories of UPSID (Maddieson and Precoda 1989).

The second aspect of inventories to be studied is the frequency of consonants across languages. Certain sounds are extremely common (such as /p/ and /m/) while other sounds are less common (such as /q/ and /h/). This frequency distribution is related to inventory size: small inventories tend to have only the most common sounds, while large inventories have all of the most common sounds, as well as rare or even unique ones (Maddieson (2011), Lindblom and Maddieson (1988), see also Section 4.2). Put another way, small inventories look similar to each other, and inventories diversify as they grow.

Lindblom and Maddieson (1988) propose that this effect is due to the way that inventories grow over time. Inventories first saturate a small set of all possible sounds, before expanding into other areas of phonetic space. A metaphorical rubber band draws inven-
tories back toward this basic set, accounting for the contents of small inventories, while a metaphorical magnet pushes sounds apart from each other, resulting in the increasing diversity of large inventories.

This metaphor is appealing but Lindblom and Maddieson do not offer any historical evidence to support it, nor do they point to any specific types of sound changes that might underlie the rubber band and magnet effect. I propose that the basis for a common set of sounds across languages is the existence of context-free sound changes, which can affect inventories of any size. Large inventories have more unique sounds because such languages also have a wider variety of phonetic contexts (due to the correlation with phonotactics discussed above).

The third aspect of inventories is something known as “feature economy” (Clements 2003, 2009). This describes the tendency for inventories to be organized around the re-use of a small number of distinctive features. For example, many languages have a six-stop system /p, b, t, d, k, g/, where the feature [voice] is re-used for contrast at each place of articulation. In comparison, a six-stop system /p, b, t, t', kʷ, k/, one that makes use of a different feature for contrast at each place of articulation, is extremely rare, if not actually unattested. Mackie and Mielke (2011) showed that natural languages exhibit higher feature economy scores than randomly generated sets of segments, but did not offer an explanation for why.

I propose that feature economy is the emergent result of sound change. This is because the phonetic biases underlying sound change are such that they can affect classes of sounds, rather than individual sounds. This makes it possible for a new set of sounds to emerge in an inventory, all of the members of which are minimally different from another set, differing only by whatever phonetic feature was affected by sound change. Over time, this creates the appearance of economy in an inventory. Randomly generated inventories are less economical than natural languages because they have never undergone sound change.

These proposals about consonant inventories will be tested through computer simulation. In Chapter 2, I present PyILM, a Python package for running sound change simulations. Broadly speaking, PyILM is an implementation of a listener-based theory of sound change. In contemporary linguistics, this approach to sound change is probably most well-known through the work of John Ohala (1981, 1983, 1991, 1997, et seq.) and more recently Evolutionary Phonology (Blevins 2004, 2006b, Blevins and Wedel 2009). The computational framework used is the Iterated Learning Model (Brighton et al. 2005, Kirby et al. 2008, Smith and Kirby 2008, Cornish 2010). This is an agent-based model where agents are arranged in a transmission chain. The nth agent receives information from agent n − 1, formulates a hypothesis about it, and then transmits new information to agent n + 1.

Sound change in a simulation is due to events known as “misperceptions” in the terminology of PyILM. Misperceptions are defined as probabilistic, context-sensitive rules (which includes context-free rules, i.e. rules sensitive to any context). When a misperception occurs, the phonetic value of a speech sound is altered. The shift in phonetic value creates the potential for the agent at generation n to acquire a different set of sound categories
compared to the agent at generation \( n - 1 \).

Simulations have a large number of parameters that can be set, which makes it possible to study how sound systems evolve under different conditions. For instance, the hypothesis that inventory size is connected to phonotactic complexity can be testing by running several simulations with identical starting conditions, varying only the syllable types permitted, and measuring the size of the inventories after \( N \) generations of transmission.

The dissertation is organized as follows: The remainder of Chapter 1 discusses issues of language transmission and sound change. Chapter 2 provides technical details of PyILM. Chapter 3 gives some toy simulations to illustrate how PyILM works. Chapter 4 returns to the topic of natural languages with an overview of cross-linguistic tendencies in inventories. Chapter 5 provides the results of PyILM simulations demonstrating how these cross-linguistic tendencies can emerge through iterated learning of sound systems.

1.2 Iterated learning models

Languages can only survive over time if they are continually re-learned at each generation. This continuation of people learning from others, who in turn learned from others, is referred to as “cultural transmission” (Brighton et al. 2005). A simulation of language change should be at least in part a simulation of cultural transmission. The actual process of cultural transmission is extremely complex, as it includes an uncountable number of interactions between an enormous network of people, often with intricate social relationships. Language use and acquisition is also tied to the physical environment, conversational context, and various other socio-linguistic factors. There may even be more than one language being transmitted at a time. This makes computational modeling of cultural transmission challenging, and it is common to abstract away from this complexity, and focus on simpler situations.

Cultural transmission is modeled in this dissertation as an Iterated Learning Model (Kirby 1998, Kirby et al. 2008, Kalish et al. 2007, Griffiths and Kalish 2007, Smith et al. 2003). This is a simple model of information transmission where individuals are arranged in a chain. The \( n \)th individual receives input from the \( n-1 \)th individual, formulates a hypothesis about the input, then uses that hypothesis to produce output for the \( n+1 \)th individual. In terms of language change, each pair of individuals is intended to represent one generation of language transmission.

In such a model, there are only ever two agents interacting at a time. There is always one agent who already knows a language, referred to hereafter as the speaker, and one agent who is learning from the speaker, referred to hereafter as the listener or the learner. These are relative terms. Every agent spends some time in both roles, with the exception of the first agent, who is seeded with some kind of language to get the simulation started, and hence is never a learner. The agent at Generation 2, for example, is a learner with respect to the agent in Generation 1, but a speaker with respect to the agent in Generation 3.
The nature of iterated learning is such that the information being transmitted can change over time. Any errors that occur in transmission can continue to get propagated through the chain of agents. It is possible that the language acquired by the final generation is extremely different from what the first generation knew. This is a desirable outcome in terms of modeling natural language change, since languages (eventually) become mutually unintelligible with their ancestral forms.

The amount of change that occurs in an iterated learning model, and how often it occurs, depends on how reliably information can be re-learned by each agent. Information that is difficult to reliably re-transmit (for whatever reason) will tend to change or disappear from a language. This is known as “selection for learnability” (Kirby et al. 2008, Brighton et al. 2005, Smith and Kirby 2008).

“In order for linguistic forms to persist from one generation to the next, they must repeatedly survive the processes of expression and induction. That is, the output of one generation must be successfully learned by the next if these linguistic forms are to survive. We say that those forms that repeatedly survive cultural transmission are adaptive in the context of cultural transmission: they will be selected for due to the combined pressures of cultural transmission and learning.” (Brighton et al. 2005, p. 10; emphasis added)

One goal of this chapter is to apply this concept to the study of sound change. This will not be difficult, since it already has much in common with popular models of sound change through misperception (e.g. Ohala 1981, Blevins 2004). The concept of selection for learnability originally grew out of research on syntax and morphology, so it is useful to start with a brief overview of that literature, even though it takes us somewhat far afield from the topic of phonological inventories. I will focus on the theoretical and computational aspects, and pay less attention to the implications for syntax.

Early work in this area was done by Simon Kirby (Kirby 1996, 1998, 2000, 2001) who has focused on how compositional syntax can emerge in a language that is initially non-compositional, through the process of iterated learning. A compositional language in this case is defined as one where the meaning of an utterance is a function of the meaning of its parts and how they are put together. A non-compositional language, on the other hand, is defined as one where every meaning is expressed through a holistic arbitrary pairing of meanings and sound strings. This is not a strictly binary division, and a language could have some meanings expressed by compositional structures while others are non-compositional. This is in fact the case with natural languages where we both observe compositional patterns (e.g. regular word order and morphological paradigms) and non-compositional forms (idioms, irregular forms).

Algorithm 1.1 gives an outline of a typical iterated learning simulation from Kirby’s work. This would simulate $g$ generations of agent interactions, and each agent learns from $d$ utterances.
Algorithm 1.1 Generalized Iterated Learning Model

Generate a speaking agent with a grammar
Generate a learning agent with no grammar
Loop $g$ times:
  Loop $d$ times:
    The speaking agent produces an utterance.
    The learning agent tries to parse the utterance with her grammar.
    If she cannot, she memorizes it as an unanalyzable whole.
    The speaker is removed from the simulation.
    The learner becomes the new speaker.
  A new learner is added into the simulation.

Kirby argues that compositionality emerges as languages adapt to a specific constraint imposed by cultural transmission, namely the fact that a learner cannot hear an example of every sentence that she will potentially want to express as a speaker. This constraint is referred to as the “transmission bottleneck”, and it is similar to the concept of the Poverty of the Stimulus in generative linguistics (Legate and Yang (2002), Berwick et al. (2011), see Zuidema (2003) for comparison of the transmission bottleneck to poverty of the stimulus). This constraint is built into simulations by ensuring that $d$ is less than the total number of utterances an agent could possibly produce.

In Kirby’s models, languages are said to “survive” transmission if the learner at generation $n$ acquires a grammar such that she would produce the same utterance for the same meaning as the speaker at generation $n - 1$. If the grammar changes between generation $n$ and $n + 1$, then the language of generation $n$ did not survive transmission. It is important to treat the word “survive” as a technical term to be understood in the context of simulations. There is a single language in a simulation, and a single pair of users at a time.

Non-compositional languages cannot survive when there is a bottleneck on transmission. This is because a learner-turned-speaker will, at some point, want to express a meaning she has never heard expressed. She will not be able to guess how this meaning would be expressed by the previous generation, due to the lack of compositionality. Instead, she will need to invent a new way of expressing this meaning, which will serve as input to the following generation. Because of this change, the older language does not survive the entire simulation. Changes like this are guaranteed to occur at each generation, due to the constant bottleneck. A slightly different language will appear at each generation throughout a simulation.

Compositional languages, on the other hand, can survive transmission even with a bottleneck. A learner need not hear an example of every single sentence. As long as a learner knows the component parts, and knows rules for putting parts together, she can construct a novel utterance that has a high chance of being the same as what the previous generation would have constructed. This increases the chances of a language surviving
transmission many generations in a row.

Kirby’s simulated agents all have the capability of learning compositional grammars, but the initial agent is intentionally seeded with a grammar that lacks compositionality entirely. The key step in the emergence of compositionality is the first time an agent invents a novel utterance. The way in which the invention algorithm works is crucial. The algorithm constructs a new utterance by looking for other meanings an agent already knows that are similar to the one she wants to express, selecting some random sub-string from there, and then adding on a new random string. The result of this invention is the introduction of evidence for compositionality into the input of the following generation. There are now two similar meanings with shared sub-strings that a learner can infer are connected by a rule.

For example, suppose Learner 1 acquired *afzaba* for “fox eats bird”. Later, this agent becomes Speaker 2 and invents *afzagatam* for “fox eats mouse” by taking the substring *afza* from a known word and adding on a randomly generated string *gatam* to the end. The following Learner 2 hears both of these utterances, and infers that *afza* means “fox eats”, *ba* means “bird” and *gatam* means “mouse”. Learner 2 could then posit a rule where either *ba* or *gatam* can follow *afza*, as opposed to memorizing each meaning independently. Learner 2 now has the first partially compositional grammar in the simulation. (See Kirby (2000) for specific details of the invention and rule-induction algorithm.)

Learner 2 will eventually become Speaker 3, and will make use of this rule to produce utterances. Any utterances that Speaker 3 invents containing the meaning “fox eats” will contain the string *afza*, and this is information that Learner 3 can use to acquire a grammar similar to Speaker 3.

Over the following generations, more and more compositional rules enter into the language through this cycle of invention and rule-induction. Since compositional languages can be learned in spite of a bottleneck, transmission has a lower error rate, and eventually the language comes to be dominated entirely, or almost entirely, by compositionality.

It is important to note that there is no single factor that can explain the emergence of compositionality. It is the result of a combination of agent behaviour and cultural transmission. Changing either of these changes the resulting languages. Trivially, if agents were cognitively incapable of using compositional structure (if they could do neither rule induction nor compositional invention) then of course compositionality would never arise. If agents were content to learn strictly from the input, and never invent new utterances, then non-compositional languages would have a higher chance of surviving.

If the transmission model did not involve iterated learning, but agents at each generation received input from the same external source instead, then invention and rule-induction would have no long-term impact, and the language would not evolve toward compositionality. The specific size of the bottleneck can change the outcome (Smith et al. 2003), potentially favouring non-compositional languages. Frequency in the input makes a difference as well, and non-compositional forms can survive if they are highly frequent (Kirby 2001).

It is also important to emphasize that compositionality appears entirely through non-
teleological means. While it is true that agents do introduce compositional utterances "on purpose" through the invention algorithm, this does not represent a teleological element of the model. The reason for inventing an utterance is not so that the language can, in the future, be compositional. Utterances are invented to solve in-the-moment needs of communication, with no regard to long-term consequences. Language change itself is not directed towards the goal of achieving compositionality, so the model is not teleological, even if the individual agent interactions could be said to have a goal. Instead, compositionality is achieved through selection for learnability.

This is not just an effect that occurs in computer simulations. It has also been demonstrated in laboratory experiments with human participants. Kirby et al. (2008) and Cornish et al. (2009) discuss experiments where compositional lexicons can emerge from an initially randomly-generated set of words through iterated learning. The first set of participants in an experiment were shown pictures of objects, each of which was paired with a randomly generated string of CV syllables. The objects were constructed out of three features: shape (square, circle, triangle), colour (red, blue, green), movement (horizontal, spiral, bouncing). Participants were not made explicitly aware of these features.

Participants were instructed to learn the names, and they were then tested on their ability to recall those names. The answers they provided in the recall test were then given as the labels for those objects to next set of participants in the experiment, and the cycle repeated. Unlike actual cultural transmission, participants in these experiments never met each other, and were not made aware that their answers would be given to other participants.

The lexicons at the end of the experiment appeared to be organized around certain features of the objects, rather than having each word arbitrarily matched to an object. Kirby et al. (2008) provide an example of a final lexicon with consistent pairing of morphemes and colours (ne is “black”, la is “blue”, ru is “red”) as well as motion (ki for horizontal movement and pilu for spiral). Shape was less consistent. Blue and back triangles were encoded as ke if moving horizontally, but as ki if moving in a spiral. Red horizontal triangles were called ke and the red spiral triangles were ha.

This is a result of the lexicon adapting to the learning requirements of the participants. It is difficult to remember nine random strings of sounds, so the first participants in the experiment tend to have a high error rate. Even if they could not remember the name of an object, they still had to supply a word of some kind for the recall test, so they invented a new word based on whatever other words they actually could remember. This immediately decreased the difficulty for the second participant, since the lexicon now contained words for similar objects that have substrings in common, which helps with learning. Some "irregular" forms still exist in the lexicon by the end, because while participants may not be able to remember nine random strings, they can remember two or three.

In summary, the idea that languages adapt to how they are being transmitted, what Brighton et al. (2005) call *selection for learnability*, is useful for understanding how patterns can emerge in languages over long periods of time. Changes happen as agents each make their own small adjustments to the language to meet their needs at a particular time. Agents
do not consider what effects their change will have on the future state of the language. Agents are not even aware that they are changing anything. They do not know what the underlying forms of the previous generation looked like, so they cannot know if they are deviating from them. Patterns tend to emerge because all agents learn under similar conditions. Changes that make it easier for one agent to learn to use the language will also make it easier for future agents to use the language, though no agents are aware of this.

1.3 Sound change models

How do phonological systems adapt to transmission? The facts relevant to the transmission of sound systems are of course very different from syntax or morphology. Learners-turned-speakers do not face the same problems with phonology as they might with syntax. Speakers may have to express novel propositions or construct unique arrangements of words and phrases, but they are never in a place where they need to invent a new sound they did not hear in their input and only rarely is there a need to construct a unique sequence of consonants and vowels.

The notion of a transmission bottleneck may still apply to some properties of sound systems, however. Stanton (2016) argues that certain patterns in stress systems are unattested because they are too difficult to learn from input data. It is also likely that the nature of the input affects the learnability of long-distance harmony patterns (McMullin 2016). As far as phonological inventories are concerned, however, the transmission bottleneck not an important factor, since it is highly unlikely that a sound is so rare in the input that a learner does not acquire it.

In order for a phonological inventory to “survive” transmission, a learner must acquire the same set of categories as the speaker. The main obstacle to successful transmission of an inventory is channel bias, which Moreton (2008, p. 87) describes as “phonetically-systematic errors in transmission between speaker and hearer, caused largely by subtle phonetic interactions”.

Moreton contrasts channel bias with analytic bias, a term he uses to refer to cognitive factors that make learning certain patterns easier or harder. Moreton argues that some patterns are not explainable without reference to analytic bias, and a complete understanding of phonology requires considering it along with channel bias. Although I agree in general with Moreton on this issue, for the purposes of this dissertation, I will focus only on the effects of channel bias. This is because analytic bias seems to be more relevant for learning phonological patterns involving the interaction of sounds. Moreton himself demonstrates the need for analytic bias by discussing vowel-to-vowel height dependencies and vowel-height-to-consonant-voicing dependencies. On the other hand, I am interested in the transmission of individual sound categories, which are more likely to be affected by channel bias.

There are many phonetic effects that could be considered channel bias. Co-articulation
can change some characteristics of a sound, such as nasal consonants taking on the place of articulation of following consonants (Kochetov and Colantoni 2011). Speakers may fail to reach an articulatory target, such as when vowels become more centralized in unstressed syllables, a phenomenon known as “undershoot” (Mooshammer and Geng 2008). Sometimes, two sounds might just be acoustically very similar and easily confusible, such as [f] and [θ] (Jongman et al. 2000).

The most important consequence of channel bias is that it introduces variability into the input of a learner. This variability is the precursor to sound change. Due to channel bias, a particular phonological category will have multiple possible pronunciations, and some of these may be different enough from each other that a learner of a language incorrectly infers that they are, in fact, representative of different categories. When the learner becomes the speaker, these different categories are re-transmitted to the following generation, cementing them into the language.

One important aspect of channel bias is that it is context-sensitive. Pronunciation does not randomly vary from utterance to utterance. The way that a particular sound category manifests itself phonetically is influenced by the kinds of sounds that occur before and after it. For this reason, a sound change such as n>k’ / _[continuant] is not expected to occur in any language because there is no obvious relationship between the continuant nature of the environment in which the change occurs, and the nasal-to-ejective change that is the outcome. It is highly unlikely that a listener would mistake [ans] for [ak’s], for example. On the other hand, a change like n>m / _p is a more natural change, because there is a phonetic connection between the environment (a labial stop) and the outcome of the change (a coronal becoming a labial).

A main claim of this dissertation is that inventories, and their typological characteristics, are the result of adaptation to channel bias over many generations of cultural transmission. In other words, channel bias is to phonological inventories what the bottleneck is to morphosyntax. The sounds that appear in an inventory are those which are the most likely to be successfully retransmitted, given the set of environments found in the lexicon, and given any channel bias that might apply in these environments. Since all humans have roughly the same articulatory and perceptual systems, it is expected that unrelated languages will be subject to the same kinds of channel bias, and hence similar patterns can arise in languages all around the world.

This type of approach to phonology is sometimes referred to as a “diachronic” approach, since the main locus of explanation is in the transmission of the language from generation to generation. In an overview article on diachronic explanations in phonology, Hansson (2008) describes them this way:

“[R]ecurrent sound patterns are the product of recurrent diachronic events (sound changes), which have their ultimate causes in the physical conditions under which speaker-listener interactions take place in language use and language transmission across generations. On this view, voicing is neutralized in
Scenario 1.

Speaker: /ut/
Listener: /ut/

distorted by vocal tract into [yt] heard as [yt]

Scenario 2.

Speaker: /y(t)/
Listener: /y/
Listener-turned-Speaker: /y/

distorted as interpreted as produced as [y(t)] heard as [y] [y]

Figure 1.1: Model of sound change through listener misperception, Ohala (1981, p. 182)

preconsonantal (as opposed to prevocalic) position not because some constraint to this effect is part of the innate endowment of humans, nor because learners are predisposed to posit only such constraints as are grounded in phonetics. Rather, languages will show some tendency to acquire such neutralization patterns for the simple reason that, in positions where distinctive voicing is hard for listeners (including learners) to detect, listeners / learners will be liable not to detect it, erroneously interpreting a preconsonantal voiced obstruent as being voiceless and encoding it as such in their mental representation of the word-form in question. If and when the pattern caused by such recurring misinterpretations becomes entrenched, the result is a language with systematic voicing neutralization precisely in those positions where such neutralization is phonetically motivated.” (Hansson 2008, pp.4-5)

The most influential line of work in this area comes from John Ohala (1981, 1983, 1991, 1997, et seq.). Ohala’s theory of sound change is based on the idea of listener misperceptions, which occur when listeners acquire something from the speech signal other than what the speaker intended. Listeners at some point become speakers, and the misperceived information serves as the basis for producing speech, which in turn becomes input for future learners. Ohala’s models have much in common with models of iterated learning. This is very clear in the diagram from Ohala (1981) shown in Figure 1.1, which predates any of the modern formal literature on iterated learning.

Note that this model of change crucially relies on a third generation. The learner
who initially misinterpreted the signal has to re-transmit this misinterpretation to a new generation. Ohala calls the change without re-transmission a “mini-sound change”. This is because:

“it would so far only involve one speaker-hearer. However, if this person’s speech is copied by other speakers, this mini-sound change could become a regular sound change, i.e. characteristic of a well-defined speech community.” (Ohala 1981, p. 184)

One of Ohala’s primary arguments is that listener misperceptions arise in the first place because speech is inherently ambiguous. Figure 1.2 is a diagram from Ohala (1997) that illustrates how stops can emerge from the ambiguity created by co-articulation. In the transition between two consonants, total or near-total obstruction of the oral tract may occur. The speaker has met the condition to produce a stop at this point, even though this was not the intention of the speaker and there is no underlying stop in their mental representation at this position in the word. Listeners may interpret this transient closure as belonging to a true stop, and assume that one really does exist in the word, leading to a sound change.

Ohala discusses several different environments where stops can appear through this process. I mention only two here. The first is when a nasal is followed by a fricative. This is noticeable in English words such as warmth, which may be pronounced as [warmθ] or [warmpθ], or length which may be pronounced as [lɛŋθ] or [lɛŋkθ].

The stop emerges as follows, using the word warmth as an example and focusing on the transition from [m] to [θ]. First the oral tract is closed at the lips for [m], but the velum is lowered. This is the initial state in Figure 1.2, where line B represents the closed lips and line A represents the open velar port.

From this initial state, the velum has to raise and the closure at the lips has to be released, with a new constriction formed at the teeth for [θ]. In between these two stages,
there is the possibility for the velum to have raised before the labial closure is released. This is the transitional state in Figure 1.2. This creates the conditions for an oral stop, and once the lips do open, there is a release of air into the fricative which can be mistaken for a stop burst. This is represented by the final state in Figure 1.2.

Ohala (1997) points to a few cases of historical change that could potentially be have resulted from listeners misperceiving the burst as being a true stop, for example, the introduction of /b/ between /m/ and /r/ in French, e.g. Latin /kamera/ > French /kumb/.

A second environment where stops might emerge is between a fricative and a lateral, e.g. [l] and [s]. During the transition between manners of articulation there may be total closure formed by the tongue against the sides or roof of the mouth. This would produce the right conditions for a [t] to be perceived. Ohala gives an example from English, where else has come to be pronounced [elts], and another example from Kwak’wala k’weltso? “to be feasted” from k’weł + so?.

Velar palatalization is another sound change that can potentially be explained through listener misperception. This is a common change k > tʃ before front vowels. Guion (1997) investigated the question of why the /k/ is fronted all the way to the post-alveolar place of articulation (see also Chang et al. (2001) and Wilson (2006)).

Guion (1997) notes that the peak spectral frequency of the burst of a velar stop is related to the frontness of the following vowel. Specifically, higher vowels result in higher burst peaks, and the peaks are highest before /i/. Guion compared these to the peak spectral frequencies of /tʃ/, which is relatively constant across different vowel contexts, but also higher than the velar peak in general. The burst of /k/ is highest before /i/, and so it is in that environment where it is most spectrally similar to /tʃ/, which is exactly the environment where the sound change tends to occur. Thus, this sound change could have its origin in misperception.

Guion conducted an experiment to further establish whether these sounds are indeed perceptually similar. Participants heard examples of a CV syllable consisting of one of [k, g, tʃ, dʒ] followed by one of [i, a, u], and were given a forced-choice identification task. As expected, [ki] was identified as [tʃi] more often than [ka] was identified as [tʃa].

James Kirby has proposed a listener-based explanation for the recent development of tone in Phnom Penh Khmer. Kirby reports that the trill /r/ is being lost in onset clusters, and is replaced by aspiration and a change in f0 contour. Kirby argues that the previous contrast of CV and CrV has transphonologized into a contrast based on f0 of the vowel. This is supported by perception experiments in Kirby (2014a) where listeners were able to use f0 as cue for distinguishing words that are underlyingly /CrV/ from those which are /CV/.

Kirby (2014b) further demonstrates how this kind of listener-led sound change can occur, this time with a series of computational simulations. Agents in a simulation receive examples of words, and their task is to assign each segment in the word to a category. Segments are classified based on four phonetic dimensions.

In addition, there is a channel bias that alters the input to agents. The bias has two
simultaneous effects: in a sequence CrV, it reduces the duration of /r/, and it lengthens the onset of the vowel. This bias has a cumulative effect, so the perceptibility of /r/ slowly decreases over the course of a simulation, while the length of the onset increases.

Early in the simulations, agents were able to distinguish /CrV/ words from /CV/ using the length of /r/, but as the simulation ran on this became impossible because of the bias. Instead, agents begin using information about the vowel onset because that information has become more salient to them, which is similar to what Kirby (2014a) describes for Khmer.

Another influential model of diachronic phonology is the Evolutionary Phonology model (Blevins 2004, 2006a, Blevins and Wedel 2009), which builds on Ohala’s work. The basic premise behind Evolutionary Phonology is that “[p]rincipled diachronic explanations for sound patterns have priority over competing synchronic explanations unless independent evidence demonstrates, beyond reasonable doubt, that a synchronic account is warranted.” (Blevins 2006b, p. 23). Common sound patterns are common because they result from common sound changes. Sound changes themselves are the result of articulatory and perceptual factors hindering perfect language transmission.

Much of Blevins’ terminology is borrowed from biological evolution, and shares something in common with the iterated learning literature, although her Evolutionary Phonology book does not cite any of that work. For instance, she has a discussion of “adaptations” (Blevins 2004, p. 54), which is reminiscent of the concept of selection for learnability, namely that sounds are selected for on the basis of their ability to survive transmission in a particular context:

“If a contrast between two sounds is just barely perceptible in a particular phonetic environment, its chances of survival in a noisy world are slight. ... In reconsidering the case of change where [ampa] is heard as [ampa] it makes very little sense to compare the sounds [n] and [m] outside the specific environment in which they occur. In the same sense that the usefulness of claws and toepads cannot be assessed outside particular physical environments in which they occur, there is no sense in which /n/ is a better or more useful nasal consonant than /m/ or vice versa. Adaptation occurs with respect to a specific phonetic context.”

Ohala and Blevins present slightly different typologies of misperception. Ohala divides misperceptions into two types, called “hypercorrection” and “hypocorrection” (e.g. Ohala (1992)). Hypocorrection occurs when a listener assumes that a phonetic effect, such as co-articulation, is an intended part of the signal, and internalizes it as such.

For example, the amount of aspiration that occurs on a stop depends on the height of the vowel that follows it (Hansson 2008, Ohala 1983). This is because in order for voicing to occur, the vocal folds need to vibrate, and this requires a suitable pressure differential between the oral cavity and subglottal cavity. During the closure phase of a stop, the pressure in the oral cavity builds to become equal with the subglottal pressure, and when the stop is released, pressure drops in the oral cavity. How fast this drop happens, and how
long it takes to achieve the right pressure differential, depends on the height of the vowel, i.e.
the size of the oral cavity through which air can escape. Higher vowels make for narrower
openings, which slows the drop in pressure, and also increases the turbulence/noisy quality
of the stop burst, which can make the stop sound more affricate-like.

If learners hypocorrect, they may infer an underlying affricate in this position, including
possibly as an allophonic variant of the stop. In fact, numerous languages have phonological
processes converting stops to affricates before high vowels, e.g. Japanese t → tf / _i

The other kind of change, hypercorrection, occurs if a listener erroneously tries to “fac-

tor out” a part of the speech signal. The main example of hypercorrection seems to be
dissimilation. For example, in Classical Greek a change has occurred such that labialized
consonants became unlabialized adjacent to rounded vowels, e.g. *luk’os < lukos ‘wolf’.
If hypercorrection is at play, then this change was caused by listeners who assumed the
labialization of the consonants was due to the adjacent rounded vowel, and removed it,
creating unlabialized consonants. Hypercorrection is expected to target phonetic charac-
teristics which have a relatively long duration (e.g. palatalization, glottalization, but not
continuancy or affrication). Unlike a hypocoercion, dissimilation and hypercorrection are
not likely to eliminate the original triggering environment for the change. This is because
the listener has to notice this environment in the first place in order to even make the
hypercorrection.

Blevins has a three-way typology of misperceptions in her model, calling them “choice”,
“chance”, and “change”. Change occurs when the phonetic signal is misperceived by the
listener due to acoustic similarities between the actual utterance and the perceived utter-
ance. For instance, a listener might misperceive a [θ] as a [f]. The misperception occurs on
the surface, and there is no correction taking place on the part of the listener. In fact, the
listener’s underlying form is remaining entirely faithful to the surface form but the surface
form was not a good representation of what the speaker intended.

Chance is a term for when the phonetic signal is accurately perceived by the listener but
is intrinsically phonologically ambiguous. The listener associates a phonological form with
the utterance which differs from the phonological form in the speaker’s grammar. Blevins’
example here is the speaker says /aʔ/ → [ʔaʔ] and the listener hears [ʔaʔ] → /ʔa/.

Choice describes a situation where there are multiple phonetic variants of a single
phonological form which are accurately perceived by the listener. The listener (a) ac-
quires a prototype or best exemplar which differs from that of the speaker; and/or (b)
associates a phonological form with the set of variants which differs from the phonological
form in the speaker’s grammar. In Blevins’ example, the speaker has an underlying form
/tuʔǝlaj/ which is variously pronounced as [tuʔǝlaj], [tuʔ’laj], or [tuʔlaj], e.g. there are
various amounts of schwa that actual appear on the surface. The listener has a choice
about whether to include the schwa in the underlying form, or factor it out as in irrelevant
transition between the glottal stop and the lateral.

Although these diachronic models tend to focus on the role of the listener, the speaker
is equally important because the speaker produces the listener’s input. The diachronic
model presented in Garrett and Johnson (2012), for example, is one that more explicitly incorporates the role of the speaker. Their model differs from Blevins and Ohala in that it attempts to categorize sound changes based on their underlying mechanisms, rather than on their outcome. Garrett et al. focus on four specific factors: motor planning, speech aerodynamics, gestural mechanics, and speech perception. Two of these, motor planning and gestural mechanics, are clearly speaker-oriented.

Certain kinds of sound changes are more obviously speaker-initiated than others. One example is assimilatory change, which occurs when there is overlap in articulation between two sounds, causing one sound to acquire the features of the other. Consider the nasalization of vowels before nasal consonants, for instance. To produce the vowel, the oral tract needs to be open to some degree and relatively free of obstruction, and there should be no nasal airflow, i.e. the velum should be raised. The postvocalic nasal consonant has conflicting requirements: the speaker needs to close off the oral tract at some place of articulation, and lower the velum for nasal airflow. Since the velum cannot be instantaneously displaced, and since oral closure cannot happen immediately, there is the possibility that the speaker will spend some time with the velum lowered and the oral tract open, which effectively results in a nasal vowel.

Vowel nasalization tends to occur more often when the nasal consonant follows the vowel, compared to when the consonant precedes it (Chen et al. 2007). This is again due to articulatory effects. When the consonant follows the vowel, the potential co-articulation happens as the speaker attempts to open the velar port and close off the oral tract. When the consonant comes before the vowel, however, the potential period of co-articulation happens as the speaker attempts to close the velar port and open the oral tract. As it turns out, the movement required for velic opening in post-vowel nasals is about 1.6 times faster than the movement required for oral opening in post-nasal vowels (Krakow (1994)). The faster speed of the velic movement means there is a greater probability of producing a nasal vowel in a VN sequence, compared to a NV sequence, because the velar port is going to open for the nasal before the oral tract can be closed.

This co-articulatory effect has been argued to be the source of historical changes where oral vowels nasalize before nasal consonants, becoming full-fledged phonemes, such as occurred in some Romance languages (Recasens 2014). It is also common for many languages to have allophonic nasalization of vowels before nasal consonants (Schourup 1973), and this too probably developed from misperceptions arising from co-articulation.

However, this articulatory timing is not universal, so nasalization of vowels adjacent to nasals is not universal either. Butcher (1999) studied the articulation of speakers of Australian languages in the Arandic, Lake Eyre, and Yura groups. He found that these speakers have systematically different timing in the raising and lowering of their velum, compared to speakers of English. In particular, the Australian speakers showed much more sudden changes in the state of their velum, which meant that nasality hardly spread at all into adjacent vowels. Butcher further suggests that this particularity in articulation is the origin of pre-stopped nasals in these languages.
The aerodynamic voicing constraint (Ohala 1983) is another example of how articulation can play a role in sound change. The constraint refers to the requirements for modal voicing: there must be air flowing through the vocal folds, which need to be tensed. This presents a problem for voiced stops. By their nature, stop consonants cause air to accumulate in the oral cavity, and the difference in air pressure above and below the glottis begins to equalize. At a certain point voicing becomes impossible. Voiced fricatives are also affected by this constraint because friction requires the air pressure in the oral cavity to be greater than atmospheric pressure. This creates a conflict: for voicing oral air pressure needs to be low, for friction oral air pressure needs to be high.

How does the aerodynamic voicing constraint factor into an explanation of sound change through misperception? The argument would run as follows: the production of any voiced stop or voiced fricative inherently puts it in conflict with this constraint. To maintain voicing, oral air pressure needs to be reduced somehow, and Ohala discusses several ways this can be achieved, including expansion of the cheeks, lowering of the larynx, or venting some of the accumulated air through the nose. On some occasions, these “strategies” can lead to speakers producing speech with characteristics different than intended, which listeners will (wrongly) assume are intended characteristics. This makes the constraint different from co-articulation, because it is not entirely context-dependent. There is an inherent difficulty in voicing an obstruent, regardless of its position in a word.

Ohala (1983) provides a list of 12 potential implications this has for sound change and inventory typology, such as the fact that voiceless stops and fricatives are more common cross-linguistically than their voiced counterparts. In P-base (Mielke 2008), for example, 97% of the languages have at least one of /p,t,c,k,q,x,s,f/ whereas only 83% of languages in the database have a voiced version of any of those.

As another example, Ohala argues that imposives developed in Sindhi from geminate voiced stops. The length of a geminate means it is even more at risk of becoming voiceless than a singleton voiced stops. Oral air pressure must be kept low for an even longer period of time through some means. Ohala proposes that this was done through larynx lowering, which listeners misinterpreted as implosion.

Another articulatory effect that plays a role is gestural reduction. Lin et al. (2014) looked at the role of gestural reduction in the production of the English lateral /l/. Alveolar laterals have two lingual constrictions, one anterior and one dorsal. The degree of anterior constriction in laterals varies with the phonetic context and between speakers, in some cases achieving no apical contact at all. In English, /l/ is especially likely to be reduced in the context of V_C, where C is non-alveolar. For instance, the /l/ in help or elk undergoes more reduction than the /l/ in melt. This may be partly due to homorganicity, and Lin et al. find that the anterior constriction is less reduced when the tongue tip would be making a contact at that place for the following sound anyway.

This reduction is one reason underlying a change currently underway in some varieties of English, where /l/ loses its anterior constriction entirely, and vocalizes to /w/ or /u/ (due to the dorsal constriction). Lin et al. report that in dialects where this change is underway,
it is more advanced in pre-labial and pre-velar contexts, which is expected given the greater likelihood for gestural reduction in that environment. The loss of /l/ has already occurred before /k/ in some words, though it is still preserved in the orthography, e.g. walk, talk, balk, etc.

1.3.1 Summary

To summarize the general model of sound change that has just been presented: Languages must be successfully transmitted from speaker to learner through the medium of physical speech in order to survive over time. There are numerous factors involved in articulation and perception, so-called channel bias (Moreton 2008), that impede successful transmission. In addition, the listener cannot necessarily know the intentions of the speaker, or if the signal has been changed in any way. This creates the possibility that learners can misperceive some aspect of the speech signal. When learners eventually become speakers, this misperceived element is then re-transmitted to the following generation, making it part of the language (Ohala 1981, Blevins 2004).

The term “misperception” in this context is intended to be neutral with respect to the actual source of the change. It could be due to perceptual or articulatory factors. The key point is that learners have acquired a language from the input that differs from the language that generated the input, but they do not realize they have done so.

The next step of the dissertation is to describe a computer simulation based on this framework of sound change.

1.4 An ILM for phonology

1.4.1 Overview

Simulating the evolution of consonant inventories requires simulating multiple, potentially interacting, sound changes over multiple generations. The following is an overview of PyILM, the simulation software designed for this dissertation\(^1\). Algorithm 1.2 represents \(g\) generations of transmission, with each agent learning from \(d\) words.

At the core, this is just a model of the transmission of sound strings. There are no morphological or phonological processes, and agents can always recover the intended meaning of a word. Full technical details, including descriptions of various algorithms, are given in Chapter 2. In the following sections, I will focus more on the higher-level conceptual details and theoretical assumptions that went into building PyILM.

\(^1\)The simulation is an Iterated Learning Model (ILM) written in the Python programming language. The name PyILM follows a convention of the Python community of prepending “py” onto the names of packages or programs. The intended pronunciation is [\textipa{pəi.əl}].
Algorithm 1.2 Generalized Iterated Learning Model for phonology

Generate a speaking agent with a lexicon
Generate a learning agent with no lexicon
Loop $g$ times:
  Loop $d$ times:
    The speaking agent produces a word from the lexicon
    Misperception may alter some phonetic values in the word
    The learning agent assigns each sound in the word to a known phonological category
    If no known categories match, then a new one is created.
The speaker is removed from the simulation.
The learner becomes the new speaker.
A new learner is added into the simulation.

1.4.2 One turn of a PyILM simulation

To understand how the simulation is intended to work, it is useful to give an overview of one iteration of the simulation. This consists of the speaker producing a word, misperceptions possibly occurring, and the listener learning something.

1.4.2.1 Production

The turn begins with the speaking agent selecting a word to produce from the lexicon. Words in the lexicon are represented as strings of phonological categories (i.e. segments). These categories are, in turn, represented as a list of binary features of length $F$. For each category (segment) in a word, a production algorithm generates an array of length $F$, where the $n^{th}$ element is a real number in [0,1], representing a phonetic value for the $n^{th}$ phonological feature. For example, assuming a very simple simulation with four features, [continuant, nasal, voice, sonorant], an instance of the category /b/ might be represented as [0.05, 0.30, 0.95, 0.04]. In an actual simulation, these numbers are determined by sampling a (truncated) Gaussian distribution for each feature. The distributions are inferred during the learning phase, except for the initial agent in a simulation, who is seeded with a set of distributions.

This process of generating lists of phonetic values is done for each sound category in the word, and then the resulting list is sent to the misperception function.

1.4.2.2 Misperception

Misperceptions in PyILM are modeled as probabilistic context-sensitive rules (which includes the null context, i.e. context-free rules). They target sounds that have particular phonological features and which exist in specific contexts, which are themselves defined in terms of features. Misperceptions may also refer to word boundaries. The effect of a
misperception is to change phonetic values. An example of a final-devoicing misperception would be represented as:

\[ [+\text{voice}, -\text{son}] \rightarrow [-.15\text{voice}] / \_\#, p=.3 \]

In prose, this reads as “on any given utterance, there is a 30% chance that voiced obstruents have their [voice] value reduced by .15 when they occur in word-final position”. The idea is that a misperception changes the surface phonetic value of a sound such that it becomes more likely the listener will categorize it as having the opposite underlying feature value.

Here I wish to emphasize again that the term “misperception” is intended as a cover term for any kind of effects that could occur during either production or perception, which might substantially affect what a learner infers about a language. Determining the origin of a sound change is important in understanding specific changes in actual natural languages, of course, but in a simulation the distinction is irrelevant. The key point is that something disrupts transmission and a learner has the potential to infer a sound that the speaker did not intend.

A list of misperceptions must be provided as input to the simulation. If none are provided, then no sound changes will occur, and simulation is pointless. The number and type of sound changes that can occur in a particular simulation, therefore, is limited. This is unrealistic, but it is an unavoidable constraint that comes with computer simulation; there must be a finite set of parameters to simulate. In any case, it is probably unfeasible to draw up a list of all possible misperceptions. PyILM could be considered an “ideal world” simulation (cf. the “ideal observer” of James Kirby (2014b)) where we know everything there is to know about what kinds of misperceptions are possible.

An alternative, and more complex, way of modeling misperceptions would be to simulate the vocal tract and auditory-perceptual systems of the agents in detail, and allow misperceptions to arise naturally from the way these systems work. Such a simulation would most certainly be useful in an effort to support the position that misperceptions arise from phonetic factors. My aim for this dissertation, however, is not to explain how or why misperceptions occur. I simply assume that they do occur, and I am interested their long-term consequences on the structure of inventories. It suffices for these purposes to simulate the effect of misperception, rather than the cause. For an example of more complex modeling of physiology, see Oudeyer (2005a, 2005b, 2005c) on the evolution of phonotactic constraints.

In other words, the sound changing rules of PyILM (“misperceptions”) are intended as useful abstractions that capture the spirit of how sound change is thought to work. They allow for a wide range of different sound changes to be simulated (including context-free ones). All elements of misperception are open to modification by the user and any number of misperceptions can be active in a given simulation run.

For example, both hypercorrection and hypocorrection can be simulated through the use of these rules. An example that Ohala (1983) gives of hypercorrection, where the
learner erroneously factors out some aspect of the signal, is the unrounding of stops before rounded vowels in Greek, /lukwos/ → /lukos/. To simulate the possibility that agents might hypercorrect in this situation, a rule such as \([-\text{continuant}, +\text{round}] \rightarrow [-.15\text{round}] / _[-\text{voc}, +\text{round}], p=.1\) would be included in the simulation.

A hypocorrection, where a learner fails to account for a phonetic effect and assumes it is inherent to the signal, would be exemplified by pre-consonantal neutralization of a voicing contrast. This could be represented in PyILM with a misperception such as \([-\text{sonorant}, -\text{vocalic}, +\text{voice}] \rightarrow [-.15\text{voice}] / _[-\text{vocalic}, -\text{sonorant}], p=.1\)

### 1.4.2.3 Learning

In the final stage of a simulation turn, the misperception function sends the list of phonetic values to the learner. The learner receives this as a list, so the ability to parse speech into segment-sized units is assumed. Learning is done using an exemplar-based model, where agents keep detailed representations of experienced events (Pierrehumbert 2001, Johnson 2007). For each sound in the word, a learner stores the phonetic values in memory, then attempts to categorize the sound by comparing these values to all other known categories. If any of them meet a threshold for similarity, then the input sound is placed in that category. Otherwise, a new category is created, the input sound becomes its sole exemplar, and future learning can be influenced by this category.

At the beginning of the learning phase, agents do not know any categories at all, so the first sound that is experienced becomes the first category, and all others are built up from there. Categorization is done on the basis of phonetic similarity. Two sounds are considered to be instances of the same category if they differ by less than some value (the particular threshold is determined by a simulation parameter that can be set by the user). At the end of learning, agents infer a Gaussian distribution from the observed phonetic values, for each feature, for each category they have created. This distribution is what will be sampled by the production algorithm when the agent becomes the speaker.

### 1.4.3 Some notes on design

#### 1.4.3.1 Social factors

PyILM is not intended to model all possible types of sound change. It focuses specifically on changes related to production and perception. Another major cause of change, which is not simulated, is contact between dialects or languages.

Contact can lead to one language borrowing words which contain sounds not found in the native inventory. For instance, some Bantu languages are known to have acquired clicks by borrowing from neighbouring Khoe-San languages (Güldemann and Stoneking 2008). Other times, however, borrowing words leads to no changes at all: English has not acquired uvular fricatives or front rounded vowels, despite borrowing numerous words from
French which contain these sounds (e.g. \textit{hors d’oeuvre}, \textit{objet d’art}, \textit{maître d’}). Instead, those words have undergone adaptation to the English sound system.

Borrowing is, in a sense, more arbitrary than sound change based on phonetic factors. Borrowing depends on coincidences of contact between cultures, and how the borrowing actually plays out depends on similarities between the sound systems of two languages, as well as various socio-linguistic factors. The frequency of borrowing, and its effect on inventories, over the history of a language is sporadic.

It is effectively impossible for languages to avoid phonetically-based change, but change through contact is avoidable. The people living on North Sentinel Island represent the extreme case of contact-avoidance. Inhabitants of the island are hostile to outsiders, and occasionally kill people who come too close (McDougall 2006). It is unlikely that the Sentinelese have borrowed many words (at least not recently), but it is quite likely that their language has undergone some kind of sound change in the last several generations.

Phonetic changes are more of a constant factor across time and across languages. They depend on factors related to human speech production and perception. We can expect phonetic factors to influence unrelated languages in similar ways, leading to cross-linguistic tendencies.

1.4.3.2 Single-agent transmission change

Another type of change not simulated is what Labov (2007) calls “diffusion”. This refers to language changes that occur when mature speakers of a language adopt the speech habits of a different group of speakers. Labov contrasts this with the term “transmission” to refer to language changes that occur as language is being passed from mature speaker to learner. PyILM is strictly a transmission chain, with a single learning agent and a single speaking agent at each generation.

Diffusion-chain versions of cultural transmission models do exist (Mesoudi and Whiten 2008, Whiten and Mesoudi 2008), and have even been applied specifically to the study of language (Smith and Wonnacott 2010). Along the same lines, Griffiths and Kalish (2007, Section 7, p.470) show how the mathematics of their single-agent model generalizes to larger populations (though this is not strictly related to diffusion chains).

Modeling transmission or diffusion requires different design choices, because the fundamental factors driving language change are different in each case. Transmission-chain models represent an acceptable level of simplification, given the goals of this dissertation. A more nuanced outcome could be achieved by combining diffusion and transmission in a single simulation. This would allow the nature of the input to the learner to vary more as the speaking agents possibly change their behaviour throughout the speaking phase.
1.4.3.3 Discrete learning period

It is common in agent-based simulations for there to be a specific learning phase, after which agents can no longer learn anything new. This approximately simulates the real-world sequence of events where one’s ability to learn language is greatest as a child, commonly known as the “critical period” (e.g. Newport et al. 2001, Pallier 2007), and slows down with age. Having a sharp, and arbitrary, cut-off point is a simplification for the purposes of computer simulation.

More broadly, research has found that the way people speak continues to change over the course of their life. For instance, Harrington et al. (2000a, 2000b, 2005, 2006) studied the Christmas broadcasts of Queen Elizabeth II taken from a period of roughly 30 years. They found evidence of a change in vowel pronunciation, in particular that the Queen’s vowels were becoming more like those of Standard Southern British speakers. In another case, Sankoff and Blondeau (2007) found a change in the pronunciations of the rhotic consonant of Montreal French, with some adult speakers moving from an apical /r/ to a dorsal /’r/.

1.4.3.4 No teleology

Sound change occurs for entirely non-teleological reasons in this model. The only thing that happens is that learners learn from the input. They make no assumptions about what sound systems should look like. In contrast, it is common in other models to give agents advance knowledge about the sound system. For example, James Kirby (2014b) describes a computational ILM for simulating a change currently underway in Khmer, where an aspiration contrast (e.g. /ka/ vs. /kʰa/) is being replaced by a tonal contrast (/ka/ vs. /ká/). The overall design of Kirby’s simulation shares much in common with PyILM, but there is a crucial difference in that Kirby’s agents are modeled as “ideal observers” which is a type of Bayesian classifier (see Geisler 2003). These classifiers require a prior probability for each phonetic category, in order to compute the probability that an input sound belongs to that category. This effectively means that agents have foreknowledge about which and how many possible sound categories could exist in the language.

Feldman et al. (2009) also use a Bayesian model for learning sound categories by learning a lexicon. In this model, learners know in advance that there are exactly 4 sound categories in the inventory. Kirby and Sonderegger (2013) consider the iterated learning of 2 vowels (not full words). Vowels are represented simply by an F1 value, and the distribution of F1 values is known to all agents, and does not change over time.

Along the same lines, in some models learning is done by selecting between a limited number of choices. Tupper (2015) gives a mathematical model for the conditions under which two vowels might merge over time, but considers only a case where agents knowingly choose between /i/ and /e/.

Wedel’s (2007) computational model has a small set of underlying sounds that map to exactly one surface sound, and the special category /x/ which has two possible allophones.
The learner must select one of the allophones as an underlying form for /x/, and Wedel showed how different types of learning error resulted in different outcomes.

Advance knowledge of possible sound categories might be useful for a simulation of the change from one specific inventory state to another, but it is undesirable for a general model of inventory change. Sound inventories need to be free to change, grow, and shrink within a relatively large space of possibilities. In PyILM simulations, there are no pre-determined sound categories at all. Agents simply build up categories based on the information available to them in the input.

In this respect, the learning algorithm has more in common with de Boer (2000, 2001, 2002). Agents in his simulations play an "imitation game", where a speaking agent produces a single vowel and a listening agent tries to imitate it. de Boer’s simulations use supervised learning, meaning that listeners are given feedback about how well they imitated, and they use this information to place prototypes into a vowel space. PyILM, in contrast, uses unsupervised learning where no feedback is given. Agents in de Boer’s simulations make no assumptions in advance about how many vowels there might be in a language. The final number and type of vowels depends on the interactions between agents and the success of individual imitations. Vowel systems ranging from three to nine vowels emerged from de Boer’s simulations.

1.4.3.5 Phonemes and allophones

Sounds are represented at two levels in PyILM. At a surface/phonetic level, sounds are vectors of real numbers. At an underlying/phonological level, sounds are lists of binary features. A simulation starts by generating a set of these categories for the first agent. First underlying categories are created, and then a distribution of phonetic values, for sampling during the production phase, is generated for each of these categories.

New sounds that are introduced through sound change are, at least initially, limited to a particular context (due to the context-sensitive nature of the misperceptions that give rise to them). These new sounds are considered as allophonic variants of whichever sound they grew out of. For example, if /b/ lenites to [v] between vowels, and this is the only instance of [v] anywhere in the lexicon, then [v] will be considered an allophone of /b/. As time goes on, these allophones eventually cease to vary with another category, and attain the status of a phoneme. This transition, from misperception to allophone to phoneme, is intended to parallel the real-world process of phonologization, where an initially phonetic effect eventually becomes a fixed part of a phonological system (Bermúdez-Otero 2007).

These categories have no bearing on the outcome of a simulation. Agents are not aware of what is a phoneme and what is an allophone, and neither the production algorithm nor the learning algorithm ever specifically reference these categories. Within a simulation, everything is considered to be a “segment”. The categorization of a sound as a phoneme or allophone is done at the end of a simulation, as a tool for understanding how sounds are distributed in a lexicon.
This issue will be discussed in more detail in Chapters 3 and 4 along with specific simulation results.

1.4.4 Expected outcomes and inventory structure

The set of misperceptions that is supplied to a simulation acts like the bottleneck in Kirby’s (1998, 2000, 2002) simulations of syntax. They are the main constraints preventing the successful transmission of sounds over time. By the end of a simulation, the expectation is for an inventory to have whatever set of sounds is least likely to be affected by these misperceptions, given the set of phonetic contexts in the lexicon.

In simple cases, it is even possible to predict what the outcome will be before running a simulation. Consider the simplest situation when only a single misperception is acting on transmission. For discussion purposes, assume there is a final-devoicing misperception which sometimes makes word-final voiced obstruents less voiced than they are in other positions. Suppose that two simulations are run, each starting with a lexicon generated from the sounds /b, d, g, i, a/. The difference now is that one simulation has a lexicon with only V or CV syllables, while the other allows up to CVC syllables.

The final inventory of the CV language is easy to predict in this case: it will be /b, d, g, i, a/, i.e. it will not have changed, since the relevant environment for devoicing does not exist in the lexicon. On the other hand, the CVC language might develop an inventory as large as /p, b, t, d, k, g, i, a/, depending on the specific contexts in the lexicon and how often misperceptions actually occurred during the simulation. For instance, if the distribution of /b/ in the initial lexicon was restricted to final position, then all instances of /b/ are prone to misperception, and there is a low probability of /b/ still existing in the final lexicon, and high probability of /p/ existing in at least one word. If no words in the initial lexicon ended in /g/, then there is no reason for /k/ to be in the final lexicon. Predicting the outcome becomes more difficult, or impossible, with a larger number of interacting misperceptions added to the simulation.

Selection for learnability occurs as sounds from the original inventory of the language change due to misperception. If a sound cannot reliably be learned, given the environments in the lexicon and the set of misperceptions acting on transmission, it will probably not survive the entire simulation. The final inventory is one that is in a sense optimized for its own transmissibility.

This simple example also demonstrates how patterns in inventories can be derived non-teleologically by modeling only individual sound changes. If we could watch the CVC language generation-by-generation, we would observe what Martinet (1952, 1955) called “gap filling”. Assume that all voiced stops appear in both initial and final position in the very first lexicon. After some number of generations, one of them would devoice due to misperception in final position, creating a stop inventory of, say, /b, t, d, g/. Later devoicing misperceptions could change the inventory to /p, b, t, d, g/, and this creates an apparent gap at the velar place (although in this hypothetical example the gap is the
opposite of the one normally found in natural languages, where /g/ is more likely to be missing). Eventually, misperception will fill in that gap for a full stop system of /p, b, t, d, k, g/, but at no point did agents intend for this to happen. They simply learn from the input, and a full suite of consonant pairs is a side-effect of the way that misperception is affecting the learning data.

1.5 Summary

This chapter introduced the concept of selection for learnability. This refers to the idea that certain linguistic patterns tend to persist over time because they are more likely to be successfully transmitted to the following generation. This concept was first developed for the study of syntax and morphology by Kirby (2000, 2001, 2002) who showed that compositional morpho-syntax can emerge over time because it is more learnable, given a limit on the number of sentences that a learner can have access to.

I proposed that phonological inventories are also selected for learnability, although in a different way. Learners are not constrained by any bottleneck on transmission (since the number of phonological categories in a language is finite and relatively small in number). Instead, the strongest influence on inventories comes from channel bias (Moreton 2008), which refers to phonetic effects like co-articulation or acoustic similarity. Channel bias introduces variability into the input of a learner, and this variability is the precursor to sound change. If learners misperceive some aspect of speech due to channel bias, then this misperception gets retransmitted to the following generation when learners become speakers.

For the purposes of this dissertation, I use the term “misperception” very broadly, and it refers to any kind of sound change that occurs because a learner infers a sound system that differs from the one used by the speakers of the language at the previous generation. This notion of misperception is in turn taken from the models developed by Ohala (1981, 1983, 1992, et seq.) and Blevins (Blevins 2004, 2007).

I implemented the basic assumptions of these diachronic models into a computer simulation called PyILM, which will be used to investigate three aspects of phonological inventories: their size, the relative frequency of their segments, and their feature economy. The following chapter provides more specific technical details of how PyILM works.
Chapter 2

PyILM

2.1 Introduction

This chapter details PyILM, a computer program written in Python for simulating language transmission, with a focus on phonology. PyILM’s design is informed by theories of sound change through misperception (e.g. Ohala (1983), Blevins (2004)), and its formal implementation is based on the Iterated Learning Model (e.g. Smith et al. (2003), Kirby (2001)). These topics were discussed in detail in the previous chapter.

PyILM simulates the transmission of a lexicon over multiple generations. It creates agents arranged in a chain, each of which learns a phonological system from the output of the previous agent. PyILM allows users to manipulate numerous parameters of this process and run iterated learning simulations to explore how sound change happens under different conditions. Section 2.2.2 of this chapter gives a complete list of the parameters that a user can set. Section 2.4 gives some more details on how to use and configure a simulation.

2.1.1 Iterated Learning Models

Computational models of iterated learning, e.g. Kirby (1999, 2001), follow this basic pattern of nested loops:

Generate a speaking agent
Generate a learning agent
Loop x times:
  Loop y times:
    The speaker produces an utterance
    The learner learns from this utterance
  Remove the speaker from the simulation
  Make the learner the speaker
  Create a new learner
This pattern represents \( x \) generations of language transmission with \( y \) learning items at each generation. The corresponding loops for PyILM are given in Algorithm 2.1. These loops are explained in “pseudo-code”, which I will continue to use throughout the chapter to explain the logic of PyILM. Pseudo-code is text consisting of valid Python expressions, most of which appears as actual lines of code in PyILM. However, some of the code has been changed to make it more readable in the context of a dissertation, hence the name pseudo-code.

**Algorithm 2.1 Main simulation loop**

1. `Simulation.load("config.ini")`
2. `speaker = BaseAgent()`
3. `Simulation.initialize(speaker)`
4. `listener = Agent()`
5. `for generation in range(Simulation.generations):`
6. `    for j in range(Simulation.words_per_turn):`
7. `        word = speaker.talk()`
8. `        word = Simulation.transmit(word)`
9. `        listener.listen(word)`
10. `    Simulation.record(generation)`
11. `    speaker.clean_up()`
12. `    speaker = listener`
13. `    listener = Agent()`
14. `Simulation.generate_output()`

Line 1 loads user-provided details about the simulation and configures PyILM appropriately. Line 2 creates a new agent for the first generation of a simulation, and Line 3 seeds it with an initial lexicon and inventory. Line 4 creates a new “blank” listener who will learn her language from the speaker. To be clear, this initialization phase is not intended to represent any actual events in language transmission. PyILM only simulates the “evolution” of language in the sense that it simulates how languages change over time; it does not simulate the emergence of language from non-language. The first speaker in the simulation represents some speaker at some point in the history of some language. Note that the first speaker is formally a different kind of object in the program than the other speakers, since the first speaker requires a set of initialization functions, while later generations rely on learning algorithms.

Line 5 starts a loop that runs once for each generation being simulated (see section 2.2.2.2). Line 6 starts a loop that runs once for each word a learner hears.

In line 7 a speaker chooses a word to say (see the production algorithm, section 2.3.4). Line 8 simulates misperception by changing some of the segments of the word (see section 2.2.10). Misperceptions are context-sensitive, and probabilistic, so sometimes nothing
at all happens on this line. On line 9 the learner learns from this new word (see the learning algorithm, section 2.3.1).

On line 10, PyILM keeps a record of what the speaker’s inventory and lexicon look like before the speaker is removed from the simulation on line 11. On line 12, the new speaker creates some probability distributions to be used during the next production phase. Line 13 generates a new listener for the next loop to start over again.

Line 14 is executed only after the final generation of the simulation has finished learning. It prints a report of what happened during the simulation, using the information logged at each generation on line 10. The program then terminates.

2.2 Objects

2.2.1 Overview

PyILM was written using an object oriented approach to programming. An “object” is way of representing a concept in a computer program. In the case of PyILM, objects represent concepts relevant to sound change or phonology. Objects have attributes which represent properties or characteristics of the objects and their values may be fixed or mutable. Objects also have methods which describe what the object can do. An example of an object is the Feature object, which represents the concept of a distinctive phonological feature. Feature objects have two attributes: name and sign. These are both strings, with name having a value of something like “voice” or “continuant” and sign having a value of either “+” or “-”. They also have an equal_to method, which is used to decide if two Feature objects are the same or not.

This example also illustrates two typographical conventions adopted in this chapter. First, the names of objects in the computer program are written with an initial capital letter to distinguish them from the use of the same words to refer to concepts in linguistics, e.g. the Feature object vs. distinctive feature. Second, the typewriter font is used when referring to object attributes and methods.

This section explains the main objects in the simulation, as well as their relevant attributes. Details about object methods are, for the most part, omitted here because they are generally not of relevance for understanding how the simulation works. One exception to this is the Agent object, which has methods for speech production and learning that are important to understanding the simulation. Examples of omitted methods include: methods for making equal to and not equal to comparisons, methods for generating string representations, and methods for reading and writing to files.

There are nine objects discussed in this section: Simulation, Word, Segment, Feature, FeatureSpace, Sound, Token, Agent and Misperception. To understand the relationship between them, it is useful to think of the objects in PyILM as being “stacked” inside one another. The diagram in Figure 2.1 is a visualization of this. Note that the figure is not
a description of object inheritance - it is a visualization of how the objects fit together conceptually.

Every run of the simulation creates a new Simulation object, inside of which there are Agent objects that talk and listen to each other. Agents all have a lexicon attribute which contains Words, which are made up of Segments, which are made up of Features.

A speaker uses a production algorithm to transform Segments into a different kind of object called a Sound, representing an actual speech sound instead of a unit in the mental lexicon. The phonetic characteristics of speech sounds are represented by objects called Tokens.

The simulation passes Sounds through a misperception function, which may alter some of their Tokens’ values, depending on the environment in which they occur. Then these Sounds are sent to the listener's learning algorithm which creates new Features and Segments.

The transmission of a single segment is illustrated in more detail in Figure 2.2. A speaker starts by selecting a word from the lexicon. The lexicon is a list of meanings, each associated with a string of segment symbols, each of which are “translated” into a set of phonological features. All possible features that can be discriminated are kept together in a FeatureSpace object, where each feature is represented as an interval [0,1]. Figure 2.2
shows only three feature dimensions (F_1, F_2, and F_3), with each dimension represented as a number line. The points circled with solid lines are the range of values that represent [-feature] segments, and the points circled with dashed lines are the [+feature] values. The lines in black represent the values a speaker experienced during their time as a learner. The lines in red represent the particular phonetic values that the production algorithm chose on this occasion. Supposing that F_1 is [voice], F_2 is [continuant] and F_3 is [nasal], then the segment is [+voice, −continuant, −nasal]. This could be represented as /b/ (among other symbols).

![Simulation diagram]

**Figure 2.2: The transmission of a phonological segment**

The values chosen by the production algorithm are then sent to the misperception function. In this example the environment was right for a misperception to occur, and the listener is going to hear the F_1 value - the voicing value - of this segment as lower than intended by the speaker.

The red lines in the learner’s FeatureSpace represent where the values were stored. In figure 2.2 the learner has already heard a number of words, and has formed some phonological categories. The boundaries will continue to shift over the course of learning.

The values for F_2 and F_3 are interpreted “correctly” by the listener, that is, her learning algorithm categorizes them as examples of the same phonological feature value as in the speaker’s FeatureSpace. Due to the misperception, F_1 is categorized as an example of the opposite feature value.
2.2.2 Simulation

2.2.2.1 Overview

Each time PyILM is run, a single Simulation object is created, and the entire simulation runs “inside” of this object. The Simulation object has methods for initializing simulation details, creating and removing agents from the simulation, causing agents to speak or listen, causing misperceptions to occur, and writing the results of the simulation to file. None of these methods are discussed here. Details about Agents (2.2.9) and Misperceptions (2.2.10) can be found in their own sections.

The Simulation object has 10 attributes representing factors relevant to cultural transmission. These are discussed here, and it is possible for users to set the value of any of these attributes. See section 2.4 for details on how to do this.

2.2.2.2 generations

The value of this attribute determines how many generations of listeners the simulation should run for. The default value is 30. This means the simulation stops after the 30th learner has finished learning.

2.2.2.3 initial_lexicon_size

This value of this attribute sets the number of words in the initial lexicon. The default value is 30. The words of the initial lexicon are created using the invention algorithm (see section 2.3.5).

2.2.2.4 initial_inventory

This attribute controls the size and contents of the initial segment inventory of the simulation. The value supplied to should be a list of segment symbols separated by commas. For instance, p,t,k,b,d,g,f,s,h,m,a,i,e would be an acceptable starting inventory. The set of symbols used should correspond with symbols in a feature file (see the features_file attribute in section 2.2.2.9).

If some degree of randomness is desired, then the value supplied to initial_inventory should be two numbers separated by a comma. The first number represents the number of consonants and the second the number of vowels. There must be at least one consonant and at least one vowel in every simulation. The segments are randomly selected from the feature file (section 2.2.2.9). The simulation determines what is a consonant and what is a vowel by checking the value of the [vocalic] feature: [+voc] segments are called “vowels” and [−voc] segments are called “consonants”.

If no initial inventory is supplied, then the default value is 10 random consonants and 3 random vowels.
2.2.2.5 minimum_repetitions

During the production phase, the speaking agent will produce every word in the lexicon at least minimum_repetition times. For example, if set to 2, then every word will produced at least twice. The default value is 1, and it cannot be set any lower.

Words in a lexicon are grouped into “frequency blocks”, with the first block containing words that are produced exactly minimum_repetition times. Each successive block that is created has a frequency of twice the previous block. Doubling the frequency approximates a Zipfian distribution of words in natural language (Yang 2010). This blocking is done randomly for the first generation in a simulation. If any words are invented during the simulation (see section 2.2.2.11) they go into the least-frequent block.

In natural language, a similar Zipfian distribution holds of individual words: the most frequent word in a language is about twice as frequent as the next one, which is is twice as frequent as the next, and so on. Early testing with PyILM found that implementing frequency distributions on a per-item basis resulted in simulations that ran for far too long. By grouping words into frequency blocks, each of which is twice as frequent as the next, the running time of the simulation is greatly improved while still maintaining a Zipf-like distribution.

2.2.2.6 min_word_length

This value sets the minimum number of syllables a word must have. The default value is 1, and it cannot be set lower.

2.2.2.7 max_word_length

This value sets the upper bound for the number of syllables a word can have. The default value is 3, and it must be equal to or greater than min_word_length. These min and max values are used by the agent’s invention algorithm when generating new words (see section 2.3.5).

2.2.2.8 phonotactics

The phonotactics attribute is used by an agent’s invention algorithm (see section 2.3.5) for creating new words. Invention happens in every simulation run to generate the lexicon of the very first agent. Invention otherwise only occurs if Simulation.invention_rate is set greater than zero (see section 2.2.2.11).

The value supplied to this attribute should be a single string that consists of only the letters “C” and “V”. This string represents the maximal syllable structure. By default, PyILM assumes that all possible sub-syllables should be allowed. For instance, if the value supplied is “CCVC”, then the set \{V, CV, CCV, VC, CVC, CCVC\} will serve as the set of possible syllables. The simulation determines which segments are consonants and which
are vowels by looking at the segment’s value of [vocalic]. [−voc] are treated as “C” and [+voc] are treated as “V”.

It is possible to exclude a subset of syllables by listing them after the maximal form separated by commas. For instance, if the string supplied to this attribute is “CCVCC,VC,VCC” then the maximal form is CCVCC, and all of its sub-syllables are allowed except VC and VCC. In other words, simulation will use the set {CV,CVC,CCVC,V}. All languages in PyILM must allow a syllable consisting of at least a vowel, so the syllable V cannot be excluded.

The phonotactics of a language are fixed for a given simulation. This is why phonotactics is an attribute of the Simulation object, as opposed to being an attribute of the Agent object. The phonotactics play a role in the outcome of a simulation (since sound-changing misperceptions are context-sensitive and phonotactics defines the set of possible contexts) so it is useful to hold them invariant for a simulation to understand their effect on sound change. The following chapter describes the output of a simulation, and there is more discussion of the specific effect of phonotactics.

The default value is “CVC”.

2.2.2.9 features_file

The value supplied for this attribute should be the name of a text file which gives a phonological feature description for possible segments. PyILM comes with a default features file that describes several hundred segments using more than a dozen features, and will be sufficient in most cases. This file is a based on the ipa2spe file available in P-base (Mielke 2008). However, users can modify this file or write their own. Each line of the file must have first a symbol, then a list of phonological features, all separated by commas. An example of such a file, with a very small feature space, is given in Figure 2.3.
i,+voice,+cont,+voc,+high
a,+voice,+cont,+voc,-high
γ,+voice,+cont,-voc,+high
i?,+voice,-cont,+voc,+high
j,-voice,+cont,+voc,+high
γ,+voice,+cont,-voc,-high
g,+voice,-cont,-voc,+high
i?,voice,-cont,+voc,+high
b,+voice,-cont,-voc,-high
k,-voice,-cont,-voc,+high
p,-voice,-cont,-voc,-high
α?,-voice,-cont,+voc,-high
f,-voice,+cont,-voc,-high
q,-voice,+cont,+voc,-high
x,-voice,+cont,-voc,+high
α?,+voice,-cont,+voc,-high

Figure 2.3: Sample feature file

The features provided in this file define the dimensions of phonetic and phonological space that can be used by a language. Every Segment in an agent’s lexicon in the simulation consists of some set of phonological features (see section 2.2.4.2). Every Sound uttered by an agent consists of a set of multi-valued phonetic features (see 2.2.7). The names of the features used in both cases are the same, and they are determined by the names in `features_file`

The symbols in this file largely serve as a kind of user interface. Sounds are actually represented in PyILM as numbers in a list, but this representation is unhelpful for a human. If PyILM has to print a symbol, for example when producing a report of the simulation, it uses these symbols as a more readable representation.

2.2.2.10 max_lexicon_size

This sets a maximum size for the lexicon of the language. If the lexicon has reached maximum size and invention is required, then one of the least frequent words in the lexicon is selected (at random) and removed from the language, to be replaced by the newly invented word. The default value is 30.

2.2.2.11 invention_rate

This represents the probability that a speaker will produce words that were not in her input. This could represent new words that have come into fashion during the speaker’s life, coinages that she created herself, borrowings, or any other source of new words. Note that
invention never introduces new sounds, so if inventions are considered to be like borrowings, then it would be a case of complete adaptation to the native sound system. Further, invented words always match the phonotactics of the language (see section 2.2.2.8). The number of invented words is determined by the `max_inventions` parameters (see section 2.2.2.12).

The value of this attribute must be between 0 and 1. If the value is set to 0, then agents never invent words and only the words used in the first generation will be the ones used throughout (though, of course, their phonological and phonetic properties may change). If the value is set to 1, then new words will join the lexicon each generation. See section 2.3.5 for more details on the invention algorithm. The default value is 0.

### 2.2.2.12 max_inventions

The invention phase only happens once for each agent, at the beginning of the production phase. Pseudo-code for this is given below. During the invention phase, the simulation will make `max_inventions` attempts to generate a new word and add it to the lexicon. The default value for this attribute is 0 (i.e., no inventions ever occur). The probability of a word actually being invented is set by the attribute `invention_rate` (see section 2.2.2.11).

```python
for j in range(Simulation.max_inventions):
    n = random.random()
    # generate a random number in [0,1]
    if n <= Simulation.invention_rate:
        word = agent.create_new_word()
        agent.update_lexicon(word)
```

In a simulation where `max_inventions` has been set to `X`, and `invention_rate` is set to `Y`, the probability that `X` new words actually are invented at a given generation is $x^y$ for $x > 0$. The probability that no new words are invented is $(1 - x)^y$. Suppose that `max_inventions` is 3 and `invention_rate` is 0.2. This means that for any generation, no more than 3 new words can enter the lexicon. The probability that only 2 new words are invented is $0.2 \times 0.2 = 0.04$. The probability that no new words enter the lexicon is $0.8 \times 0.8 \times 0.8 = 0.512$.

### 2.2.2.13 misperceptions

This should be the name of a text file, in the same directory as PyILM, that contains a list of misperceptions that could occur over the course of a simulation. Each line should have six arguments separated by semi-colons. The PyILM Visualizer also contains an option for creating misperception files with a more intuitive user interface. See section 2.2.10 for more details on Misperception objects.

The first argument is the name of a misperception. The second argument is a list of phonological features, separated by commas, that describe segments that can be altered by
the misperception. The third argument is the name of a feature which undergoes change if the misperception occurs. Multiple values for the third argument are not permitted, and misperceptions can only change one feature at a time. The fourth argument is a number in the interval (0,1) representing how much the phonetic feature changes if the misperception happens. The fifth argument is the environment in which the misperception can occur. Environments should be specified at the level of phonological features. A special value of * can be used to mean “context-free”. The sixth and final argument should be a number in (0,1) representing the probability that the misperception occurs. The default set of misperceptions are in file called “misperceptions.txt” that is bundled with PyILM and users can also consult this file for an example of the format to follow.

| pre-nasal nasalization bias; nasal,+voc;nasal;05; _+nasal,-voc;15 |
| word final devoicing; +voice,-nasal; voice;1; _#; 2 |
| intervocalic lenition; -cont,-son; cont;1; +voc _+voc;1 |
| default vowel voicing; +voc; voc;05; *; .75 |

In the first example, the misperception targets non-nasal vowels. The nasal value of these vowels is raised by 0.05 when these vowels appear before nasal consonants, and on any given utterance where such a context appears there is a 0.15 chance this happens. The second example targets voiced non-nasal segments. These sounds have their voicing values decreased by 0.1 when they appear word finally, and this happens with a 0.2 probability.

The final example shows a context-free misperception, which is referred to as a “bias”. In this case, vowels have their voicing values raised slightly under all circumstances. This allows vowels to be affected by the final-devoicing misperception, but not to the same extent as consonants because their voicing value gets raised a little bit anyway. (Of course, vowels could be completely excluded from the final devoicing misperception by changing the targeted segments to be include [−voc] only.) In Chapter 5, the effects of misperceptions and biases are explored in more detail.

The amount by which a misperception changes a sound cannot be greater than 1 or less than -1. This is because phonetic values in PyILM must be numbers between 0 and 1. If the effect of misperception would push a value above 1, then PyILM will force the value back down to 1. Similarly if a misperception were to push a value below 0, PyILM will raise the value back up to 0.

2.2.2.14 minimum_activation_level

This attribute should be a number in [0,1] representing how close to an existing category an input examplar must be, in order to be considered for membership in that category. Technical details of the examplar learning algorithm are given in section 2.3.1.
Setting this number closer to 1 sharpens an agent’s discrimination, and permits smaller distances between categories. This leads to inventories with more segments and less variation in pronunciation. Setting it all the way to 1.0 means that an input exemplars only count as a member of an existing category if they have phonetic values that match exactly to all other exemplars in the category. This is a rare occurrence, so inventories tend to grow rapidly with this setting.

Setting this number closer to 0 blurs an agent’s discrimination and increases the distance required between categories. Setting it all the way to 0 means that there is no distance between categories, and every input sound after the first counts as an example of whatever the learner first heard. This leads to an immediate collapse in the segmental inventory and it reduces to a single segment within a generation.

2.2.2.15 auto_increase_lexicon_size

The attribute initial_lexicon_size (see section 2.2.2.3), is used to determine the number of words in the lexicon of the initial generation. These words are all randomly generated, using the set of sounds supplied to the initial_inventory (section 2.2.2.4) attribute. However, in this randomness, it sometimes happens that not all of the initial sounds actually make it into a lexical item. If auto_increase_lexicon_size is set to True, then PyILM will continue to generate words for the initial lexicon until every sound occurs at least once, even if it means surpassing initial_lexicon_size. If auto-increasing is set to False, then the lexicon size remained capped at the the initial value.

2.2.2.16 initial_words

This parameter allows the user to submit a list of words, separated by commas, that should appear in the lexicon of the initial generation. It is the user’s responsibility to ensure that the words contain symbols which appear in the initial lexicon of the language. If a word supplied to initial_words contains a symbol not in the initial inventory, then PyILM will raise an exception and stop running. In short, this parameter cannot safely be used in combination with a randomly selected started inventory (see section 2.2.2.4).

It is possible for a user to create words that do not conform to the phonotactics of the language with this parameter, although this is not recommended as it may cause unexpected behaviour during the simulation.

The initial lexicon is guaranteed to include every word supplied to initial_words, even if this means going beyond the lexicon size supplied to initial_lexicon_size (section 2.2.2.3). If this occurs, PyILM will also not enforce the auto_increase_lexicon_size (section 2.2.2.15) parameter, and the initial inventory will consist only of the sounds found in the initial_words list. If, on the other hand, the number of initial words is smaller than the initial lexicon size, PyILM will continue to randomly generate words until the lexicon is an appropriate size, and the
auto_increase_lexicon_size parameter works as expected. By default, this option is not turned on and the initial lexicon will consist of randomly generated words.

2.2.2.17 allow_unmarked

Normally, sounds in a PyILM simulation are represented as lists of binary features, marked as either [+feature] or [−feature]. If the allow_unmarked option is set to "True", then a third feature value "n" is allowed (this is the "unmarked" value). If a sound is marked [nfeature], it means that every instance of that sound experienced by a learner had a phonetic value of 0 on a given feature dimension (see sections 2.2.5 and 2.2.6 for more details on how features work in a simulation). In practice, [nfeature] would be used in cases where a feature does not apply at all to a sound, e.g. a glottal stop could be marked [nlateral] because the tongue body is not involved whatsoever in the articulation of a glottal stop.

Note that [nfeature] is not equivalent to [−feature], even though sounds with either feature value will have low phonetic values on particular dimension. If the allow_unmarked option is used, it is important to ensure that misperceptions are formatted properly (see section 2.2.10), and specifically make reference to [nfeatures] if desired.

The default value of allow_unmarked is "False", meaning that only binary features are used in a simulation. Every simulation reported in this dissertation was run with the default value for this option.

2.2.2.18 seed

This attribute controls the random seed used in PyILM. Its value can be any number or string. By default is it a randomly selected integer in [1,10000]

2.2.2.19 seg_specific_misperceptions

This parameter used to create a special kind of misperception, for the purposes of a testing a hypothesis about feature economy to be presented later in this dissertation. Details are given in Chapter 5, Section 5.4. This parameter takes a value of either True or False, and the default is False.

2.2.3 Words

Words are generalized objects that represent either an entry in a mental lexicon or an utterance of one of these lexical items. Words have two attributes: string, which is a list representing the segmental melody of the word, and meaning, which is an integer.
2.2.3.1 string

If a Word is in a lexicon, then the string attribute is an ordered list of Segments (see section 2.2.4) representing the segmental content of a word, plus two word boundary symbols “#”. The value of string in this case is learned, and updated, as part of the learning algorithm (see section 2.3.1).

If a Word represents an utterance, then string is an ordered list of Sounds (see 2.2.7). In this case, the value of string is determined by the production algorithm (see section 2.3.4).

2.2.3.2 meaning

The meaning attribute is just an integer. The meaning attribute in used when the learning algorithm checks if an input word means the same thing as a known lexical item. Two words are considered to “mean the same thing” if their meaning attributes compare equal. This only very roughly models the concept of “meaning”, and is certainly not representative of what a real human speaker knows about the meanings of words. This simplification is sufficient for the purposes of modeling sound change.

The first word invented in the simulation is assigned a meaning of 0. Then a counter is started, and each new word is assigned the next integer. One consequence of this process of generating meanings is that there can be no poly-morphemic words in any language. Every word has a single meaning, so any language generated by PyILM is completely isolating.

Another consequence is that no synonyms will ever appear in the language. If a listener hears two words that mean the same thing, it will always be two instances of the same word. The counter doesn’t run backwards, so speakers will never invent a new way of saying something they can already say. There can be variations in the pronunciation of a word, due to phonetic effects or misperceptions, but there can be no completely unrelated forms with the same meaning.

2.2.4 Segments

Segments represent mental categories, themselves representing speech sounds, that agents can learn. Segments are the units that make up the words in an agent’s lexicon. They are not the actual speech sounds that agents produce and perceive. In other words, they are like phonemes, not phones. The objects representing speech productions are called Sounds, and they are described in section 2.2.7.

Segments are not atomic objects. They are represented by a set of phonological features (see 2.2.5). Features in turn are abstractions representing a range of values along a particular dimension of phonetic space.

The ability to segment speech is taken for granted in PyILM and learners are assumed to be able to portion out the speech signal in such a way as to form some kind of segment-like unit. The particular phonetic and phonological characteristics of segments are, of course, learned during the simulation and not pre-determined.
Segments have three attributes: symbol, which is an identifier for the segment, features, which is a list of distinctive phonological features, and envs, which lists all the environments the segment appears in.

2.2.4.1 symbol

Segments all have a symbol attribute, which is a string of Unicode characters (normally but not necessarily of length 1). The symbols are drawn from those provided to the simulation’s features_file variable (see section 2.2.9).

Symbols are just a convenience for the simulation, and the actual choice of a symbol can be entirely arbitrary. However, experience has found that choosing segment labels at random makes it very difficult to interpret the simulation results. The natural intuition of a linguist is to assume that IPA characters are used meaningfully, so if instead they are randomly associated with features, then reading simulation results becomes a frustrating puzzle of trying to remember what, e.g. /p/ stands for this time. Instead, PyILM tries to choose a “reasonable” symbol for a given segment by selecting a symbol for a segment whose feature description most closely matches the segment under consideration. This means that in most cases, the segment symbol will match the phonetic values in a way that makes linguistic sense, although it is always safer to inspect the actual feature values and not rely on the symbol.

In some cases, sounds can appear in a simulation that have a feature specification not found in the user’s feature file. For instance, a user might include a misperception that nasalizes stops under some conditions. Normally, stops are [−sonorant] and nasals are [+sonorant], but this nasalization misperception can create sounds that are [−sonorant, +nasal]. PyILM needs a symbol for such a sound, and if it cannot find a perfect match in the user’s feature file, it will take a sound that matches most of the features. This can occasionally lead to unexpected results when visualized (e.g. PyILM might pick a symbol for a plosive to represent the non-sonorant nasal). Again, it is always safer to check feature values than to rely on the assigned symbol, especially in simulations with a large number of misperceptions and less-predictable outcomes.

2.2.4.2 features

This attribute is a list of distinctive Features (see 2.2.5) that uniquely characterize the segment. These values are inferred by the agent’s learning algorithm (see section 2.3.1), and may change over the course of learning. They are fixed after learning ends, and do not change once the agent becomes the speaker. The actual distribution of phonetic values corresponding to a particular segment is recorded in a FeatureSpace object. These objects are described in more detail in section 2.2.6.
2.2.4.3 envs

The `envs` attribute keeps track of all the environments in which a segment appears. The `environment` of a Segment is defined as the immediately adjacent Segments to the left and right. More formally, the environment of Segment in position $j$ of a Word’s string is a tuple consisting of the Segment at position $j-1$ and the Segment at position $j+1$. Words in an agent’s lexicon begin and end with word boundaries, and these provide the appropriate environment for word-initial and word-final segments. Although word boundaries are formally treated as Segments, they differ from the Segments described in this section in that they lack phonological features and they are not considered part of an agent’s inventory. They only exist as part of an agent’s lexicon; the objects transmitted to learners do not contain word boundaries and must be re-constructed by the learner. This has no practical effect on the simulation at the moment because speakers only ever utter one word at a time.

2.2.4.4 distribution

The distribution attribute is a normal probability distribution representing the distribution over possible phonetic feature values for the segment. An agent keeps in memory one distribution for each feature dimension. These are created at the end of the learning phase.

2.2.5 Features

A `feature-dimension` is a one-dimensional space, a number line, representing some salient, gradable property of speech that listeners are aware of and can use to categorize speech sounds. Feature objects represent phonologically significant ranges of values, and are created by the learning algorithm (see section 2.3.1). These ranges represent values for phonological features, and there are three possible feature values: "+", "-", "n". The [+feature] category is for ranges of higher values, and the [-feature] category is for ranges of lower values. The actual values depend on the details of the simulation in question. A third value is [nfeature] which is assigned to segments that are always expressed with a value of 0 for the feature (e.g. a glottal stop would be [lateral]). By default, the “n” value is not used in simulations, and all features are binary. To use it, set Simulation.allow unmarked to True in the configuration file.

The number of feature dimensions is determined by the features listed in `Simulation.features_file` (see 2.2.2.9). The clustering into features is also illustrated in figure 2.2 on page 30.

Feature objects have two attributes: `sign` and `name`. The `sign` attribute can have value of "+" or "-", (and possibly “n”) and `name` is drawn from those provided to `Simulation.features_file`. The value of “n” is only available if the `allow_unmarked` option is turned on (see section 2.2.2.17). This option is turned off by default.

The set of feature-dimensions is fixed at the beginning of a simulation and does not change throughout (see section 2.2.2.9). This set is represented as a FeatureSpace object
(see section 2.2.6). Having a fixed set of features does not, of course, mean that every one
of them will participate in a contrast for a given simulation. For instance, it is possible
for a language to have no lateral consonants in the initial generation, meaning no segments
marked [+lateral], and to not acquire any over the simulation run. So long as every sound
has relatively low values along the [lateral] dimension, they will all get classified as [−lateral],
and the feature will not be contrastive.

2.2.6 FeatureSpace

A FeatureSpace object is a multidimensional space representing all possible phonetic values.
Every utterable sound in the simulation can be represented by some point in this space.
FeatureSpaces have \( f \) feature-dimensions, where \( f \) is the size of the set of features provided
to the `Simulation`:`features_file` attribute (see section 2.2.9). Points in any given
dimension fall somewhere in the interval \([0,1]\). Formally a FeatureSpace is just a Python
dictionary (a hash table) where the keys are the names of a feature-dimension and values
are lists of Token objects (see section 2.2.8) that are stored along that feature-dimension.

This has the consequence that one phonological feature in a language corresponds to
only one kind of phonetic feature along a single dimension. This is unlike natural language
where a range of phonetic characteristics may be related to a phonological feature. For
instance, [voice] may correspond with VOT, burst amplitude, and F0 (e.g. Lisker (1986),
Raphael (2005)).

2.2.7 Sounds

Iterated learning models place a heavy emphasis on the fact that language exists in both a
mental form and a physical form. The Segment objects discussed in section 2.2.4 represent
part of mental language. The corresponding object representing physical speech is called
a Sound. The difference between a Segment and a Sound is analogous to the difference
between a phoneme and a phone. Sounds are created by the production algorithm (section
2.3.4), further manipulated by the misperception function (section 2.2.10), and serve
as input to the learning algorithm (section 2.3.1).

Sounds as objects are similar to Segments. Sounds only have `symbol` and `features`
attributes, although `features` is a list of Token objects (see section 2.2.8), rather than Feature
objects which is the case with Segments. A Sound exists for only a single event of trans-
mission, then is removed from the simulation. The environment in which a Sound occurs
is calculated in place by the misperception function, or the listener’s learning algorithm, as
needed.

2.2.8 Tokens

Tokens represent the spoken values of Features (section 2.2.5), much like Sounds (sec-
tion 2.2.7) represent spoken Segments (section 2.2.4). Features actually represents ranges
of phonetic values, and a Token object represents one value from that range. A Token has
four attributes: name, value, label, and env.

2.2.8.1 name

The name attribute is the name of whichever phonetic feature this Token represents (see
section 2.2.2.9).

2.2.8.2 value

The value attribute is a number in [0,1]. In a sense, value represents how “strongly” a
given sound expresses a particular feature (whichever feature is given for the name attribute).
Larger values are intended to represent increasingly salient or prominent information, al-
though what this means would depend on the feature in question. What makes something
more [nasal] in actual speech would depend on, e.g. nasal airflow, degree of closure, nasality
of adjacent segments, etc.

2.2.8.3 label

The label attribute is a reference to the symbol attribute of one of the Segment objects
in the agent’s inventory, indicating that a Token counts as an exemplar of that particular
segment. The values along a given feature dimension that are associated with a particular
segment will change over the course of learning, and Token labels are continually updated
to keep in line with changes to the inventory.

2.2.8.4 env

The attribute env represent the environment in which the Token was perceived. This
consists of a tuple of two references, the first a reference to the Segment on the left and the
second a reference to Segment on the right. These references allow a dynamic updating
of the FeatureSpace as the learner’s inventory changes over the course of learning. For
instance, suppose a learner has perceived a word-initial Token before a vowel which gets
labeled /e/. The env attribute of this Token would be the tuple (#,e). If the category /e/
later gets merged with another vowel category and has its label changed, perhaps to the
label /i/, then the env of this Token would be automatically updated to (#,i).

2.2.9 Agents

Agent objects represent the people who learn and transmit a language. Agents have four
important attributes: lexicon, inventory, feature_space, and distributions. There
are also three Agent methods described in their own section: a production algorithm (2.3.4),
a learning algorithm (2.3.1), and an invention algorithm (2.3.5).
In addition to the Agent object, there is also a BaseAgent object, which is used for agents in the 0th generation of the simulation. The two objects share many attributes and methods, and formally speaking Agent inherits from BaseAgent. These distinctions are largely unnecessary for understanding how the simulation works, however, so they are ignored here and I present all of the relevant information under the general heading of “Agent”.

2.2.9.1 lexicon

Entries in the lexicon are represented by another kind of object called a Word (see section 2.2.3). Words are learned and stored in the lexicon as part of the learning algorithm (see section 2.3.1). Lexicons are essentially just “storehouses” of words. The lexicon is a mapping between meanings and lists of Words that can be used to convey that meaning. Each possible Word is stored alongside the raw count of how many times it appeared in an agent’s input. Multiple Words can become associated with the same meaning through misperception. For instance, in a simulation with a final devoicing misperception, the meaning 17 might be associated with the list /pad (6), pat (4)/, which would indicate that [pad] was heard 6 times during an agent’s learning phase, while the word [pat] was heard 4 times. Put another way, meanings are analogous to lexical items, and Words are phonological representations of these lexical items.

2.2.9.2 inventory

The inventory of an agent is a list of all the Segments that appear in at least one Word in the agent’s lexicon. The inventory is used in learning to make comparisons between words (see section 2.3.1). The inventory is also used by the invention algorithm (see section 2.3.5), which creates new arrangements of known segments.

2.2.9.3 feature_space

This attribute just serves to point to a FeatureSpace object (see section 2.2.6). This object represents a multidimensional phonetic space, and every sound that an Agent can hear or produce is represented as a point in this space. The feature_space of an Agent is initially empty, and it gets filled with points, which are then clustered, during learning. The production algorithm makes use of these clusters for deciding on what phonetic feature values to assign to different phonological values.

2.2.9.4 distributions

An agent’s distributions attribute is a dictionary organized first by segment, then by feature. Each feature is mapped to a probability distribution, which is sampled by the
production algorithm when it needs phonetic values. This is described in more detail in section 2.3.4.1.

2.2.10 Misperception

The idea behind misperceptions is that some sounds, in some phonetic environments, are susceptible to being perceived by the learner in a different way than the speaker intended. For instance, there is a tendency for word final voiced obstruents to be pronounced in such a way as to be perceived as voiceless (Blevins (2006b)). This can lead to instances of misperception where a speaker intends /bab/ and the learner understands /bap/.

Misperception objects are intended to represent factors inherent to human communication that affect perception of sounds probabilistically, in well-defined environments. These are factors that could potentially affect speech perception at every utterance, and so become relevant to the cultural transmission of sounds. For instance, speakers produce oral vowels with more nasality before nasal consonants (Chen (1997)). This fact about the pronunciation of vowels in certain environments means that in the transmission of any language with words that contain a sequence of an oral vowel followed by a nasal consonant, there is some small probability that learners will mistakenly interpret these vowels as inherently nasal, leading to a sound change where vowels articulated as oral vowels at one generation are articulated as nasal vowels in a later generation (cf. Ohala 1983)

On the other hand, Misperception objects are not intended to represent instances of misperception caused by e.g. the conversation happening at a loud concert, or peanut butter in the speaker’s mouth. These factors certainly affect production and perception of speech sounds, but they do not occur with enough regularity to be worth including in a simulation of cultural transmission.

Formally speaking, a misperception is a probabilistic, context-sensitive change to a Token object’s value attribute. Here are two examples:

\[+\text{vocalic}] \rightarrow [\text{nasal} + .1] / \_ [\_ \text{nasal}], .2 \text{ ("pre-nasal nasalization")}\n
\[+\text{voice}, -\text{son}] \rightarrow [\text{voice} -.15] / \_ \#, .3 \text{ ("final-devoicing")}\n
The first example reads as “There is a .2 chance that Tokens representing [+vocalic] Segments have their [nasal] value increased by 0.1 if they occur in the environment before a Segment marked [+nasal]”. The second example reads as “There is a .3 chance that Tokens representing voiced obstruents have their [voice] value decreased by 0.15 if they occur in word-final position”.

The probabilities are arbitrary and chosen for illustration. The probability of any misperception actually occurring is an empirical question, and not one that PyHLM can be used to answer. Instead, users can set this value and run multiple simulations to understand how higher and lower values affect the overall course of sound change. In fact, all aspects of a misperception are open to modification by the user (see the Simulation.misperceptions attribute, section 2.2.2.13).
Misperceptions have six attributes: **name**, which identifies the misperception, **target**, which describes the segment susceptible to misperception, **saliency**, which is a number representing units of change, **env** which describes when the change happens, and **p**, which represents the probability of a change happening. These are described in the subsections below and section 2.2.10.7 gives the pseudo-code for how misperceptions are handled in PyILM.

### 2.2.10.1 name

The **name** attribute is a string used to refer to the Misperception. It has no role in the outcome of a simulation. In fact, its only use is to print the report at the end of a simulation. PyILM lists misperceptions that applied during the simulation so that users can more easily understand why certain sound changes happened. Keeping this in mind, **name** should be something descriptive, such as “pre-nasal vowel nasalization” or “final obstruent devoicing”.

### 2.2.10.2 target

The **target** attribute is one or more phonological features representing the class of sounds affected by the misperceptions. In the case of final devoicing, this attribute would probably be set to “+voice, -son, -voc”.

### 2.2.10.3 feature

This attribute is the name of the feature that changes if the misperception occurs. In the case of final devoicing, the value of this attribute would be “voice”. The **feature** attribute is often, but not necessarily, one of the features listed in the **target** attribute. Only one name is allowed for this attribute.

### 2.2.10.4 saliency

The **saliency** attribute represents the magnitude of a change caused by misperception. The attribute can be any real number in \([-1,1]\). If a misperception actually happens, then its saliency is added directly to the value of the affected Token (see section 2.2.8). However, Token values must remain in the range \([0,1]\). If the saliency would drive a Token’s value beyond those bounds, the value is rounded back to 0 or to 1.

### 2.2.10.5 env

The **env** attribute is a string representing the environment in which a misperception takes place. There are three possible formats for this string: “X_”, “_X”, “X_Y”, where X and Y are strings consisting of the names of one or more features separated by commas, and the
underscore represents the position of the sound that might be misperceived. For instance, the following are acceptable values:

+voice_

-nasal_

_ + voice,− son

+voc_+voc

2.2.10.6 p

The p attribute is a number in (0,1) that represents the probability of a misperception occurring.

2.2.10.7 How misperception happens

Misperception applies to the output of the production algorithm (see section 2.3.4, see also figure 2.1 on page 29). The output of the misperception function is sent as input to the learning algorithm. This means there are no further changes that can apply to any sounds in a word, once the word is received by the learning algorithm.

The misperception function loops through the utterance, and checks to see if any of the segments are in a position where they might be affected by misperception. This done by comparing the environment of that segment to the `env` attribute of the Misperception. If they match, then PyILM “rolls the dice”, so to speak, and there is some probability, based on the Misperception’s p attribute, that change happens. The pseudo code for this is given below.

Algorithm 2.2 Misperception function

```
1 for sound in utterance:
2     e = Simulation.get_environment(sound, utterance)
3     for mis in Simulation.misperceptions:
4         if Simulation.check_for_misperception(mis, e):
5             # if a misperception is applicable in this environment
6             if set(mis.filter).issubset(set(sound.features)):
7                 # and if it is applicable to this sound
8                     if random.random() <= mis.p:
9                         # and it applies on this occasion
10                             sound.features[mis.target] += mis.saliency
11                             # change the feature values of the sound
```
2.2.10.8 A note on misperception definitions

The way that misperceptions are defined can affect the outcome of a simulation. Misperceptions target phonological features. What this means is that when a misperception has had its full effect, and a sound has switched categories, then the misperception will stop applying. For example, suppose that a simulation has a word-final devoicing misperception that targets sounds marked [+voice, −son, −cont], and suppose further that /b/ appears word-finally in the initial generation. Eventually a word like /ab/ will become /ap/. At this point, the devoicing misperception no longer applies to the final consonant, because it has become [−voice] and the misperception targets only [+voice] sounds.

This is a design choice for PyILM, and it is not a claim about the way that sound change operates. It is certainly not the case that the phonetic effects underlying sound change suddenly stop occurring just because of the way that some people have organized their mental grammar. The idea in PyILM is that after a sound has changed categories (e.g. from [+voice] to [−voice]), then it is irrelevant if any further phonetic effects occur. If a speaking agent has recategorized /b/ as /p/ then it does not matter if final devoicing applies to /p/ any more, since it is already voiceless.

If, on the other hand, the final devoicing misperception had been designed to target only [−son, −cont] sounds, without reference to [voice], then even after the switch from /b/ to /p/ the misperception will continue to apply. This leads to a “polarization” effect, where the phonetic values for a sound influenced by misperception will continue to get pushed to the extreme ends of a feature dimension. For example, tokens of a sound affected by the final devoicing misperception will eventually all have a [voice] value of 0. (This may also cause a further recategorization if the allow_unmark option is enabled; see section 2.2.2.17.) A misperception that raises a feature value will likewise eventually push all tokens of a category to have a phonetic value of 1.

For this dissertation, all misperceptions were designed in such a way as to avoid the polarization effect.

2.3 Algorithms

This section details three kinds of algorithms used by agents in the simulation: learning, production, and invention.

Information about an agent’s phonological system is represented using an exemplar model (Johnson 2007, Pierrehumbert 2001). These are models of memory where learners keep “copies” of every experienced speech event. These copies are known as exemplars. Exemplars are stored in a multidimensional space, and can in principle be stored at any level of detail. In PyILM, this space is a FeatureSpace object (see section 2.2.6), and the exemplars are stored at the level of phonetic features as Token objects (see section 2.2.8). The number of dimension in this space is equal to the number of features listed in Simulation.features_file.
Both the learning and production algorithms are influenced by the exemplar model. The learning algorithm works by comparing input values to the exemplars in memory. The production algorithm generates phonetic values from a distribution that is created based on the exemplar space.

This section on Algorithms is organized from the perspective of a newly created agent in the simulation. The first thing an agent does is learn, followed by an update of their lexicon and inventory, and finally an agent reaches the production phase.

2.3.1 Learning algorithm

There are two phases to the learning algorithm: parsing and updating. In the parsing phase, the learner assigns a category to each incoming sound. The results of categorization are used in the second phase to update the lexicon and inventory. After these steps have been run for each input word the simulation runs a phonological feature clustering algorithm.

2.3.1.1 Parsing a Word

The goal of this phase of learning is to assign each Sound of the incoming word to a Segment category. This is done by comparing the phonetic similarity of the input with all the previous inputs that are stored in memory. If the input is sufficiently similar to any segment in the learner’s inventory, then it is assigned to that category. Otherwise, a new category is created. The overall learning process for a word is described in Algorithm 2.3.

Learning starts with the input of a Word object (see section 2.2.3) consisting of Sounds (see section 2.2.7). Sounds have a features attribute, although this actually consists of Token objects (see section 2.2.8), not Feature objects. Tokens have phonetic values, which are real numbers in [0,1]. For each phonetic dimension, the new token is first stored into the exemplar space. Then it is compared to every other token in the space, and an activation value is returned for each such comparison.

The activation function referenced on line 5 is based on Pierrehumbert (2001). It is described in Algorithm 2.4. Each of the existing categories is assigned an “activation” value, with higher activation values representing greater phonetic similarity. Activation of an exemplar is measured as $e$ raised to the power of the negative difference between the input token and the exemplar. Activation of a segment category is the sum of the activation of its exemplars.

Agents have a threshold for similarity, which is controlled by a simulation parameter called minimum_activation_level (see section 2.2.14). It is a number in [0,1] that represents the degree to which a segment category must be activated in order for agents to consider an input token to be a member of that category. A value of 0.8, for example, means that in order for a input token to be considered a member of an existing segment category, the averaged activation of all exemplars for that category must be 80% of the maximum possible activation.
Algorithm 2.3 Learning algorithm

def learning(input_word):
    best_matches = list()
    for sound in input_word:
        activation_matrix = dict()
        for token in sound.features:
            activations = calculate_activations(agent.inventory, token)
            for seg, value in activations.items():
                activation_matrix[seg].append(value)
                #activation_matrix[seg][j] equals #how much seg is activated
                #on the jth feature
            for seg in activation_matrix:
                total = sum(activation_matrix[seg].append(total))
                activation_matrix[seg].append(total)
            activation_matrix.sort(key=lambda x:x[-1])
        best_matches.append(activation_matrix[-1])
        #best_matches[j] equals #the category with the highest activation
        #for the jth position in the word
    category = None
    new_word = Word()
    for seg, activation in best_matches:
        if activation >= 0:
            category = agent.inventory[seg]
        else:
            category = create_new_category(seg)
        new_word.string.append(category)
        agent.update_feature_space(category)
    agent.update_inventory(new_word)
    agent.update_lexicon(new_word)
Algorithm 2.4 Activation function

1. `def calculate_activation(input_token):
2.     actual_activation = sum(math.e**(-1*(input_token - exemplar))
3.         for exemplar in feature_dimension)
4.     min_activation = sum(math.e**-(1-Simulation.
5.         minimum_activation_level) for exemplar in
6.         feature_dimension)
7.     distance = scipy.integrate.quad(lambda x: math.e**-x,
8.         min_activation, actual_activation)
9.     return distance

The activation function uses this `minimum_activation_level` parameter to calculate the specific minimum activation level for the given feature dimension and segment category. Then it calculates the difference between the actual activation and the minimum by treating these values as points on the curve $y = e^{-x}$ and calculating their distance. If this distance is greater than or equal to 0 the input token meets the similarity threshold for this segment category (at least on this feature dimension) and might be considered as a potential match. If the distance is a negative number, then the actual activation level is lower than the minimum and the input token is not similar enough to this segment category on this dimension.

These distances are returned to the main algorithm, and they are summed and added to the activation matrix. Then the distances on each phonetic dimension are summed, and if any of these total distances is greater than or equal to 0, then the input token is assigned to the category with the highest value. Otherwise a new segment category is created.

After learning, another algorithm searches for any “spurious” categories that might have been created. A spurious category is one where the interval of exemplar values representing the category are a sub-interval of some other category, along every dimension.

Spurious categories crop up early in the learning phase when the exemplar space is still sparsely populated, and they do not occur in every learning phase. To illustrate this, consider the following hypothetical simulation where the speaker’s inventory has two fricatives /s/ and /z/. For simplicity, assume there are only three features: [nasal, continuant, voice].

The speaker produces an example of /s/, which has values [0.01, 0.9, 0.1], i.e. the sound has low nasality, high continuancy, and low voicing. This is the first sound the learner has heard, and it is assigned to the category labeled /s/, which matches the category in the speaker’s inventory (although the learner does not know this, of course).
Then the speaker produces an example of /z/, with values [0.02, 0.8, 0.6]. This is nearly identical to /s/ on the nasality and continuancy dimensions, but differs quite a lot on the voicing dimension. Assume that in this case this difference is sufficient for the learner to decide that this sound is not an example of /s/. Since /s/ is the only category the learner knows yet, a new category has to be created for this new sound, and it is labeled /z/ (whether the learner actually does make a new category depends on the value of minimal_activation_level, see section 2.2.2.14).

Next, the speaker produces another example of /s/, this time with values [0.01, 0.85, 0.35]. This sound is similar to both /s/ and /z/ on the nasality dimension, and relatively close to both on the continuancy dimension. On the voicing dimension, the new sound is quite distant from both /s/ and /z/. Assume the learning algorithm considers this sound to be close to neither /s/ nor /z/, and assigns to its own category, labeled /Z/ (again, in a simulation this would depend on minimal_activation_level). This category will become the spurious one. By the end of the learning phase, the range of values that the learner associates with /Z/ are going to be indistinguishable from those associated with /s/, since both of these sets of values were drawn from the same underlying distribution, namely the one the speaker associates with /s/.

As learning progresses, the learning agent hears more and more examples of the fricatives, and the exemplar space begins to fill up. For the purposes of this example, suppose that by the end of learning, there are exemplars of /s/ with nasality values ranging from 0 to 0.03, continuancy values ranging from 0.8 to 1.0, and voicing values ranging from 0.05 to 0.4. If the exemplar(s) associated with the (spurious) category /Z/ were to be fed back into the learning algorithm at the end of the learning phase, they would surely be categorized as /s/.

To check for spurious categories, an algorithm does a pairwise comparison of every segment in the inventory. For each pair of sounds A and B, it checks if the minimum exemplar value of sound A is greater than or equal to the minimum value sound B, and also if the maximum value of sound A is less than or equal to the maximum value of sound B, across every feature dimension. If both conditions are true, on every dimension, then sound A is considered to be spurious. In this case, all exemplars labeled A are relabeled B, and A is removed from the inventory.

2.3.1.2 Creating new segment categories

When an input Sound has phonetic values that are too different from any known category, a new Segment object is created. Its Token values are analyzed, and phonological features are assigned, as described in section 2.3.3. A symbol is then chosen for the segment based on these phonological feature values.

The symbol is chosen in a fairly simplistic way. The program consults the possible segments in the list provided to Simulation.feature_file (see section 2.2.2.9), and assigns a score to each of them by comparing distinctive feature values. A symbol scores 1 point
per feature match. The highest ranked symbols are put in a set and the rest discarded. This remaining set is further filtered to remove any symbols that are already in use in the inventory. A symbol is randomly chosen from the remaining set members. A random selection of this sort is safe, since the symbol has no effect on the outcome of the simulation, and exists purely to increase the readability of the output.

2.3.2 Updates

2.3.2.1 The lexicon

Once the input word has now been transformed into a list of Segment objects, the learner can add it into the lexicon. If a word with this meaning has never been encountered before, the agent creates a new entry for this meaning in her lexicon, and adds the input word with a frequency of 1.

If this meaning has been encountered before, the agent checks to see if this particular pronunciation is known, i.e. checks to see if there is a match between the input Word’s string attribute and the string attribute of any Word already in the lexicon. If so, the frequency of that Word is increased by 1, if not the input Word is added to the list of possible pronunciations with a frequency of 1.

2.3.2.2 The inventory

In the final phase of learning, the inventory is updated. This may involve one of two things. If the input word contained phonetic values such that a new segment was created, then the inventory needs to have that segment added. Even if the input word matched entirely to known segments, the specific values associated with each of those segments must now be updated.

2.3.3 Determining phonological feature values

Phonological categories are determined using a k-means algorithm that clusters exemplar values along each feature dimension.

The algorithm begins by creating k points in the space representing the objects that are being clustered. These initial points are called “centroids” and they represent a potential center of a cluster. Then every point in the data is added to the cluster with the closest centroid. After all the data has been classified this way, new centroids are chosen by calculating an average point for each of the existing clusters. The data is then reclassified by clustering it based on the new centroids. This process of averaging and reclassifying is repeated until the point where the algorithm chooses the same centroids two loops in a row.

In the case of the simulation, the clustering function takes two arguments: feature, which is the name of the feature dimension to cluster, and k, which is the number of clusters. By default k=2, since phonological features are typically modeled as binary.
Algorithm 2.5 K-means algorithm

```
1 def kmeans(feature, k=2):
2     old_centroids = agent.learned_centroids
3     new_centroids = [random.uniform(0.1, 0.25), random.uniform(0.75, 0.9)]
4     while not old_centroids == new_centroids:
5         clusters = dict()
6         tokens = agent.feature_space[feature]
7         for token in tokens:
8             closest = 999
9             for c in new_centroids:
10                if abs(token.value - c) < abs(token.value - closest):
11                    closest = c
12         clusters[closest].append(token.value)
13         old_centroids = new_centroids
14         new_centroids = list()
15         for k in clusters:
16             new_centroids.append(sum(clusters[k]) / len(clusters[k]))
17         agent.learned_centroids = old_centroids
18     return clusters
```
On Line 4 the algorithm chooses two centroids that are relatively far apart from each other, which is typical for the initial choice of centroids. Then the main loop is entered on Line 5. The loop exits when the choice of centroids doesn’t change across loops. Then from Lines 6-12, the tokens for a given feature dimension are selected, and for each token, it is compared to the most recently selected set of centroids, called new_centroids. On the first loop these are random choices, on further loops they are calculated averages.

The clusters dictionary assignment on Line 13 creates a mapping from centroid values to a list of Tokens assigned to that centroids cluster. Then the new_centroids values are saved into old_centroids and the new_centroids is emptied out (Lines 13-15). Finally, on Lines 16 and 17, new_centroids is filled with new centroid values calculated as the average of the Tokens in the clusters dictionary.

The loop then returns to the beginning. If the most recent run of Line 17 calculated the same average values as the last time Line 17 was run, then the loop breaks and the program jumps to Line 18 where agent saves the values from new_centroids and the function returns the new cluster centroids.

There are two possible phonological feature values: + and -. After the k-means clustering is done, the cluster with the higher centroid is designated the [+feature] cluster, and the one with the lower centroid is designated the [−feature] cluster. If all tokens fall into a single cluster, then a feature value is chosen based on the values of the tokens. If most of the tokens have a value above .5, then [+feature] is assigned to the entire cluster, otherwise [−feature] is assigned. If the allow_unmarked option is set to True (see section 2.2.2.17), and there is a category where every token value is 0, then [nfeature] is assigned.

2.3.4 Production algorithm

The production algorithm selects a Word from an agent’s lexicon to produce, and transforms each of the Segments in the Word into a Sound. While Segments in an Agent’s lexicon are made up of phonological features, Sounds are made up of phonetic Tokens which have real-valued features. There are three steps in production described here.

2.3.4.1 Initialization

This step occurs at the end of the main simulation loop, just after a learner has been “promoted” to speaker (see 2.1). The new speaker looks through their inventory, and for each segment it estimates a distribution of phonetic values along each dimension. Agents assume the distribution is Gaussian. Pseudo code is given below. This code runs once for each segment in an agent’s inventory. During testing, it was found that the distributions were better estimated using distance from the median, rather than from the mean. The Gaussian distribution is implemented using the normalvariate function of Python’s built in random module.
**Algorithm 2.6** Distribution estimation

```python
def estimate(segment):
    for feature in segment.features:
        cloud = {token.value for token in agent.feature_space[feature] if token.label == segment.symbol}
        median = agent.calculate_median(cloud)
        mad = agent.calculate_median([abs(value - median) for value in cloud])
        agent.distributions[segment][feature] = random.normalvariate(median, mad)
```

### 2.3.4.2 Step 1: Word selection

Production begins with a decision: select a word from the lexicon or invent a new word. The probability with which a new word is invented is given by the simulation's `invention_rate` attribute (see section 2.2.2.11). If a new word is required, the speaker uses the invention algorithm described in section 2.3.5 to create one. Otherwise, one is chosen from the lexicon.

Rather than choosing a word directly, agents actually first select a meaning, then choose which word to produce for that meaning. Each meaning in the lexicon is associated with a list of Words, each stored alongside a raw count of how many times it appeared in the input. The production algorithm chooses a Word with probability proportional to it is frequency.

### 2.3.4.3 Step 2: Transforming Segments into Sounds

The word selected by the first step consists of Segments (see section 2.2.4), but these are not the objects transmitted to the learner. In the second step of the production algorithm, Segments are transformed into different objects known as Sounds (see section 2.2.7), which represent an instance of a segment being pronounced. Agents pass through each feature of each segment in the word. For each feature, agents sample a value from the appropriate probability distribution for that segment. Pseudo-code for this algorithm is shown in Algorithm 2.7.
Algorithm 2.7 Production algorithm

1. def produce(lexical_item):
2.     utterance = Word(list(), lexical_item.meaning)
3.     for segment in lexical_item:
4.         sound = Sound()
5.         for feature in segment.features:
6.             phonetic_value =
7.                 agent.distributions[segment][feature].sample()
8.                 sound.features[feature] = phonetic_feature
9.                 utterance.append(sound)
10.    return utterance

The utterance returned at the end of the algorithm represents what the speaker intends to produce for the listener. It is not necessarily what the listener hears. This utterance is subsequently sent through a misperception algorithm (see section 2.2.10) which may change the utterance in some way.

2.3.5 Invention algorithm

The invention algorithm serves two purposes. It is used to generate a lexicon for the initial generation of the simulation, and it is used by agents at later generations if they choose to create a new word. The words constructed by this algorithm always conform to the existing syllable shapes of the language. The following pseudo-code outlines the algorithm.

Algorithm 2.8 Invention algorithm

1. def invent(agent, phonotactics):
2.     word = Word()
3.     syl_length = random.randint(Simulation.min_word_length,
4.                                    Simulation.max_word_length)
5.     for j in range(syl_length):
6.         syl_type = random.choice(Simulation.phonotactics)
7.         for x in syl_type:
8.             if x == 'V':
9.                 seg = random.choice(agent.inventory.vowels)
10.                else if x == 'C':
11.                   seg = random.choice(agent.inventory.cons)
12.                word.string.append(seg)
13.    return word
Line 4 creates a new ‘empty’ word object (see section 2.2.3). Line 5 randomly determines the length of the word in syllables (see section 2.2.2.6 and 2.2.2.7).

The loop that begins on line 8 will run once for each syllable in the word. Line 10 selects some possible syllable for a given phonotactics. For example, if (C)V(C) was supplied to the algorithm, then it selects randomly from the set \{CVC, CV, VC, V\}.

The loop that begins on line 9 runs through each segment in the syllable type chosen. For each C or V “slot”, PyILM randomly selects a segment of the appropriate type. Once the entire word has been constructed, it is assigned a new meaning and then stored in the lexicon.

The invention algorithm does not check to see if there is an existing word with the same segmental material as the new one. In other words, it is possible for homophones to appear in the language. However, this has no effect on production or learning of these words, so it is basically irrelevant to the outcome of the simulation.

2.4 Using PyILM

2.4.1 Obtaining PyILM

The source code for PyILM is available for download from https://www.github.com/jsnackie/PyILM. It is recommended to run PyILM using Python 3.4. There are also some 3rd party libraries needed: Numpy and SciPy are necessary to run the basic PyILM code, and the Visualizer requires Matplotlib and PIL (Python Image Library). All of these can be obtained from the Python Package Index at https://pypi.python.org.

2.4.2 Configuration files

PyILM simulations require a configuration file. These files should be saved into a folder called ‘config’, which must be a subfolder of the main PyILM directory. A configuration file is a text file which must conform to a particular structure, described below, and its file extension must be .ini. Configuration files are broken up into sections, each indicated by a header in square brackets. Each line in a section may contain a parameter name followed by an equals sign followed by a value. (This is the standard INI file format used on Windows.) An example is given in Figure 2.4, with some discussion following.

There are four section headers recognized by PyILM: [simulation], [misperceptions], [inventory], and [lexicon]. The order of the parameters in a section is not important. The [simulation] section is mandatory. The parameter names which can be used are listed in Section 2.2.2. Any parameter that is not mentioned in the configuration file will be given a default value. These defaults are likewise described in Section 2.2.2.

The [misperception] section is also mandatory. Each line in this section can include the name of a misperception as a parameter (any name is allowed, and spaces are permitted),
[simulation]
initial_lexicon_size=30
generations=30
phonotactics=CCVCC
invention_rate=0.05
minimum_repetitions=2
min_word_length=1
max_word_length=3
[misperceptions]
"misperceptions"
stop lenition =-voc,-cont,-son;cont;5;+voc_+voc;25
nasalization =-cont,-son,-nasal,-voc;nasal;5;_+nasal,+son,-voc;25
initial fortition =-voc,+cont,-nasal;cont;5;#_;25
stop aspiration =-voc,-son,-voice,-cont;hisubgl;5;_+voc,+high;25
obstruent glottalization =-voc,-son,-cont,-voice;mvglotcl;5;_voc,+glotcl,-mvglotcl;25
"biases"
ejectives are marked =-voc,-son,-cont,+glot_cl,+mvglotcl;mvglotcl;-.1;*;5
retroflex is marked =-ant,-distr,-cont,+cor,-son,-voc;ant;1;*;5
[inventory]
start =p,t,k,b,d,g,m,n,f,s,z,a,i,u
#start =10,3
[lexicon]
words=kapa,mufu,tiki,matk,bziafn

Figure 2.4: Example configuration file
and the remainder of the misperception’s details follow the equal sign. See section 2.2.10 for more information on how to structure a misperception.

The [inventory] section is optional, and only allows the single parameter name start, which takes the same possible values as the [simulation] parameter initial_inventory (see section 2.2.2.4). The [inventory] section exists to make it conceptually easier to manage the inventory separately from the other simulation parameters, and because future versions of PyILM are anticipated to have more possible parameters in this section.

The [lexicon] section is also optional, and can take the single parameter words, which is the same thing as using the initial_words parameter in the [simulation] section (see section 2.2.2.16).

The [simulation] and [misperception] sections should come first in a configuration file. The [inventory] and [lexicon] sections, if present, should come at the end.

If a line in the file begins with either the symbol “#” or “;” then PyILM will ignore the entire line. This can allow users to include comments to themselves about parameters. It also provides a convenient way of flipping parameter values between simulations without keeping multiple copies of a configuration file with minor differences. The use of the # symbol is demonstrated in Figure 2.4 where the words “misperceptions” and “biases” are included as comments, and there is an alternative possible starting inventory. If these symbols are encountered in the middle of a line, they are treated normally, which is what allows misperceptions to make use of both symbols without any problems.

2.4.3 Running a simulation

There are two ways to run simulations. From a command line, navigate to the PyILM directory and then type

```
python pyilm.py filename
```

where filename is the name of your configuration file. If no filename is provided, then all the defaults are used.

To run a simulation from within another python script, use the following code, replacing the string “config.txt” with the name of the appropriate configuration file.

```
import pyilm
sim = pyilm.Simulation('config.txt')
sim.main()
```

There is also a secondary program that can be downloaded for running multiple simulations, called pyilm_batch.py. To run a batch of N simulations, type the following in a command line

```
python pyilm_batch.py filename N
```

where filename is again the name of a configuration file. Supply the string None for the filename to use all defaults. For example, to run a batch of 25 simulations from within a Python script, use the following code:
import pyilm_batch
batch = pyilm_batch.Batch('config.txt', 25)
batch.run()

When running in batch mode, the user-supplied value for the random seed is ignored, and a different random seed is generated for each simulation, while keeping all other configuration details the same.

2.4.4 Viewing results

After running the first simulation, PyILM will create a new a folder called “Simulation Results”, which will be placed in the same folder as pyilm.py. Each simulation is given its own subfolder inside of the Simulation Results folder. These subfolders are named “Simulation output (X)”, where X is a number automatically assigned by PyILM.

This output folder contains a copy of the configuration file, as well as files detailing the state of the simulation at the end of each generation. Information about the exemplar space is written to files with the name feature_distributionsX.txt where X is the generation number. Information about the inventory and lexicon is written to files with the name temp_outputX.txt, with X again standing in for a generation number. It should be noted that PyILM starts counting at 0, not 1. Generation 0 is the initial generation seeded with information from the configuration file. Generation 1 is the first generation to learn from the output of another agent. A side-effect of this is that the first simulation you run will be in the folder “Simulation output (0)”.

The output files can be opened and inspected, but they are not formatted to be human-readable. They are intended for use with the PyILM Visualizer, which is an independent program that displays the information in a graphical interface. As such, it is not recommended that you change any of the names of the files, or alter any of the contents, because this can cause unusual behaviour in the Visualizer.

The Visualizer can be opened by double-clicking the file visualizer.py which comes with PyILM. It will be located in the same folder as the main PyILM program. When the program launches, select the Data menu, then input the simulation and generation number that you wish to see. Blank lines are interpreted as the number ‘0’. From there, it is possible to navigate between simulations and generations using the “Forward” and “Backward” buttons on the top right, or by returning to the Data menu.

Each generation shows the segment inventory as a table of buttons. Clicking on a button brings up more details about that segment, including its distribution in the lexicon, phonetic and phonological properties. More information about the simulation can be viewed under the Synchrony and Diachrony menus. Synchrony options include anything specific to the generation currently displayed, such as the lexicon. Diachrony options include the ability to plot changes over time. Misperceptions, which do not change, are listed under Synchrony.
2.5 Other notes

2.5.1 Limitations

PyILM cannot do everything. The program is designed largely to explore the long-term consequences of misperception-based sound change for segment inventories. There are several other ways in which sound systems can change over time that are not modeled.

2.5.1.1 No social contact

One of the limitations of PyILM is that there is only ever a single speaker and a single listener, so sound changes that rely on contact between speakers of different languages is not possible.

   Human cultures speaking different language often live nearby and interact with each other. This often leads to languages borrowing words or morphemes from the other language. Occasionally, entire paradigms are borrowed. This can lead to changes in a sound system if the borrowed items contain phonemes that are not part of the borrowing language. For example, click consonants have entered into some Bantu language through borrowing (Güldemann and Stoneking 2008). There is no guarantee of this occurring, of course, so
it is also quite common for languages to change the sounds of loanwords so that they fit native patterns (Peperkamp 2004).

The focus of the dissertation is on how phonetic effects influence the evolution of sound inventories, so no borrowing is simulated. It would be possible, however, to implement a simple form of borrowing in PyILM with a few additions. At arbitrary points in a simulation, generate new words that contain one or more sounds not figuring already in the simulation, and add them to the speaking agent’s lexicon. Loanword adaptation can be simulated by running these words through a speaking agent’s learning algorithm to see which categories any novel sounds might be assigned to.

2.5.1.2 No deletion or epentheses

Changes are also limited to those that affect feature values. Deletion and epentheses do not occur. The main reason for excluding these changes is because they can change the phonotactics of a language, and phonotactics will play a relevant role in the simulations reported later in this dissertation.

In fact, deletion is technically possible in PyILM, but simply not implemented for any simulations that I report for the dissertation. Epentheses is considerably harder to implement, and is currently not possible.

Suppose that we want to implement an epentheses rule that inserts a vowel between two non-continuants. The effect of the epentheses rule should be that a phonetic vowel appears; there is no underlying vowel in the lexical item that corresponds to the epenthized vowel. Suppose it is a mid-central schwa-like vowel. Because it is a phonetic epentheses, we cannot simply use a schwa symbol - it must be represented by a column of numbers. How do we generate these numbers?

There are three options for this. One is to generate numbers for a mid-central vowel based on the speaker’s exemplar space. This is easy if the speaker happens to have such a vowel already in their inventory. If there is no such vowel, then it is difficult to come up with a general solution for which other vowel would be the “closest”, since any arbitrary vowel system is possible in PyILM. In any case, whatever vowel is chosen will not be a mid-central vowel, so it will not correspond to the description of the epentheses rule. This makes the behaviour of the simulation unpredictable from a user’s perspective, and is not a good design choice.

The second option of generating numbers using the listener’s exemplar space has the same problems. It is further complicated by the problem that their exemplar space continues to change throughout the learning phase so the type of epenthized vowel would, again, vary unpredictably.

The third option is to include in a PyILM a generic vowel “generator” that can be used to epenthesize a vowel of a predictable quality in every case. This option feels extremely artificial compared to the first two, where at least there was some semblance of changes being related to either articulation or perception. On the other hand, it does make it easier
to follow the changes that occur over the course of the simulation.

2.5.1.3 No morphology or syntax

Words always convey a single meaning, and agents never utter more than one word at a time, so there is effectively no morphology or syntax in the simulation. Since some sound change might emerge from interactions at word or morpheme boundaries, this limitation does prevent modeling certain kinds of change. However, the changes that occur at a morpheme boundary are essentially of the same type as change that might occur within a morpheme. The root cause of the change is still a phonetic interaction of two adjacent sounds.

2.5.1.4 No long distance changes

Misperceptions that occur in PyILM can only target adjacent sounds. It is not possible to simulate the emergence of any types of harmony patterns, for example. Although consonant harmony is rare, it does exist, and plausible historical routes for its development have been proposed (e.g. Hansson 2007). However, the types of consonants that emerge from long-distance changes are a subset of those that might emerge from local changes. Since the goal of this dissertation is to understand how inventories change over time, there is no particular gain to be made by including long-distance changes.

2.5.2 Running time

The running time of a simulation is determined by a number of different factors.

The most important are the `lexicon_size` and `min_repetition` parameters. Together, they determine the total number of words that a speaking agent will produce in a given generation. If there is no maximum lexicon size, and `invention_rate` > 0, then running time can increase for each generation if new words are added to the lexicon.

Another factor is word length, since PyILM has to check every segment in each word for possible misperceptions, and the learner has to analyze each segment. The parameters controlling word length are, of course, `min_word_length` and `max_word_length`. Phonotactics also plays a role here too, since the average word length in a CV language is going to be shorter than the average word in a CCVCC language, other things being equal.

The number of misperceptions seems to have no significant effect on total running time. Checking if a misperception applies is trivial and, in most cases, nothing happens. The number of contexts where misperceptions apply is much smaller than the total number of contexts in the entire lexicon. When a misperception does apply, the operation is, again, trivial since changing phonetic values consists of adding two numbers together, followed by a check to ensure no phonetic value goes below 0 or above 1.

A single generation of a CV language with 30 initial words and a maximum lexicon size of 30 takes less than a second. Setting the phonotactics to CCVCC increases the time
significantly, and a single generation may take 10 seconds. The recording phase, where PyILM generates an output file for use with the visualizer, also contributes to running time. The length of time it takes for a generation to be recorded depends on the number of changes that occurred in the simulation.

Another factor is the time taken in labeling segments for human readability. During the simulation, segments are simply numbered, rather than being assigned IPA symbols. This is because there is no way to know which symbol will be appropriate until the end of the learning phase, when the phonological feature values are assigned. Searching the list of all possible symbols, and comparing feature values to see which would be best, can be time consuming. PyILM looks for a short-cut by comparing against the previous generation, and where sounds have not changed feature values it simply re-uses the old symbol.
Chapter 3

Sample simulations

3.1 Introduction

In this chapter I will give some examples of simulation output, using relatively simple parameter settings. This will help to clarify how the various parameters contribute the outcome of a simulation, and how various historical changes can be simulated.

Configuration files will be presented throughout this chapter. They are presented as tables, rather than being formatted as actual configuration files, for purposes of readability. Similarly, parameter names in these tables have been somewhat changed to employ regular typeface and formatting conventions, e.g. max_lexicon_size is written here as Max lexicon size. Not all simulation parameters are indicated, due to the large number of parameters. Each simulation is presented to illustrate a point, and only parameters relevant to the topic under discussion are indicated. The features file used is the default one, which is based on the Sound Pattern of English (Chomsky and Halle 1968) feature specifications available in P-base (Mielke 2008).

3.2 Simulation 1 - A single abrupt change

This first simulation is very simple. The configuration file for the simulation is show in Table 3.1.

The initial inventory is obviously not a natural inventory, but by keeping it artificially small, it is easier to understand what is happening. There is only a single misperception that can occur, final devoicing, and there is intentionally a single voiced stop /b/, so that only one sound is susceptible to change in the simulation.

The misperception is shown in Table 3.1. The “target” column shows the features that a sound must have in order for the misperception to apply. The “feature” column shows the feature which changes if the misperception applies. The “salience” column shows the direction and magnitude of a change. The “environment” column shows the context where
a sound must occur in order for the misperception to apply. Finally, the “probability” column shows, of course, the probability that the misperception applies. The salience of the change in this case, .5, makes it very likely that a learner will assign tokens affected by the misperception to a different category than those not affected. In other words, it makes it likely that sound change will happen.

Phonotactics and word length are tightly regulated so that all words will have VC or V shape. This is an extremely unnatural pattern not found in human languages, so this is for the purposes of illustration (although see Breen and Pensalfini (1999) for an argument that Arrernte is a language without onsets). The phonotactic settings ensure that /b/ will occur only in final position, which will in turn guarantee that the misperception occurs at some point. The lexicon of the initial generation will be limited and repetitive, consisting of 30 random draws from the set {iq, is, ib, i}. Figure 3.2 shows how the inventory of the language changes over the course of the simulation. Segments shown in parentheses are allophones (the precise meaning of “allophone” in the context of PyILM will be described below).

Table 3.1: Configuration for Simulation 1

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>inventory</td>
<td>b,n,q,i</td>
</tr>
<tr>
<td>Lexicon size</td>
<td>30</td>
</tr>
<tr>
<td>Generations</td>
<td>10</td>
</tr>
<tr>
<td>Phonotactics</td>
<td>VC</td>
</tr>
<tr>
<td>Invention rate</td>
<td>0</td>
</tr>
<tr>
<td>Min word length</td>
<td>1</td>
</tr>
<tr>
<td>Max word length</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Misperceptions</th>
<th>Target</th>
<th>Feature</th>
<th>Salience</th>
<th>Environment</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final devoicing</td>
<td>+voice, -son, -cont</td>
<td>voice</td>
<td>-0.5</td>
<td>π</td>
<td>0.25</td>
</tr>
<tr>
<td>Generation 0</td>
<td>Labial</td>
<td>Coronal</td>
<td>Dorsal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>b</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 1</th>
<th>Labial</th>
<th>Coronal</th>
<th>Dorsal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>(p) b</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 3</th>
<th>Labial</th>
<th>Coronal</th>
<th>Dorsal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>p</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 4</th>
<th>Labial</th>
<th>Coronal</th>
<th>Dorsal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>p</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>(b) n</td>
<td>q</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of inventories in Simulation 1

Note that all simulations start with a generation 0, which is the initial generation that is seeded with the information from the configuration file. The inventory of generation 1 is the first that could potentially have undergone sound change. As Figure 3.2 shows, right away in generation 1 some of the tokens of /b/ have been misperceived as belonging to a different category /p/ and a voiceless stop has entered the language. Initially, this /p/ is just a variant of /b/. Certain words in the lexicon are always pronounced as [ib], while others vary between [ib] and [ip]. Eventually, after enough generations of the simulation, a few words come to be pronounced as [ip] all of the time. At this point /p/ is no longer just a variant of /b/, and is now a full member of the inventory of the language.

PyILM keeps track of how sounds are changing in this respect. In the Visualizer the "total" inventory is a count of the number of sounds that occur anywhere in the lexicon. The "core" inventory is the set of sounds that all occur in at least one word where they do not vary with anything else. In generation 1 in Figure 3.2, the total count for the inventory is given as 5 (four total consonants plus one vowel), while the core count is given as only 4 (three core consonants plus one vowel), since at this point in the simulation /p/ occurs only as a variant of /b/. In generation 3, the core count rises to 5 as there are now some words with a /p/ that does not vary with /b/. Sounds in the core inventory are also shown in the Visualizer with raised button backgrounds, while the variants are shown with sunken button backgrounds.

This is analogous to the distinction between "phonemes" and "allophones" in phonological theory: the core inventory is all the phonemes, and the total inventory includes both the phonemes and all of their allophones. More specifically, a phoneme in PyILM is any sound that occurs in at least one word where it does not vary with any other sound. An allophone is a sound that occurs uniquely as a variant of another sound. I will continue to use the terms phoneme and allophone throughout this chapter as convenient labels for
these types of simulated sound categories, but with the understanding this is not the usual sense of these terms.

In particular, it is normal to define phonemes in terms of contrast: sounds that contrast with each other (i.e. participate in minimal pairs) are assigned to different phoneme categories, while sounds that do not contrast (either due to complementary distribution or free variation) are analyzed as allophones of a single phoneme. Minimal pairs or overlapping distribution are not necessary for phonemic status in PyILM.

The initial lexicon of a simulation is generated to include minimal pairs, but it is not always possible to ensure that every phoneme has a minimal pair with every other. This is because there also is a parameter controlling for the size of the initial lexicon, and the number of minimal pairs required for all sounds to have a pair can exceed the lexicon maximum. In this very small example, there are in fact many minimal pairs in the initial lexicon because it only consists of the words {ib, in, iq, i}. Additionally, when /p/ enters the core inventory, it immediately participates in a minimal pair with all the other consonants, making it more obviously a new phoneme in the language. With larger inventories, larger lexicons are required to get the full number of possible minimal pairs.

Another common criterion for determining allophones is complementary distribution. This is usually balanced with a requirement that the allophone be phonetically similar to the underlying phoneme category, since accidental complementary distribution can occur, e.g. in English [h] is only ever in initial position and [ŋ] is in non-initial position, yet these are not considered allophones of the same underlying category. Neither of these criteria are considered for determining allophones in PyILM. This is largely due to the difficulty of implementing algorithms in PyILM that can accomplish this. It is not impossible - there do exist algorithms for estimating the probability that two sounds are allophones. For example Peperkamp et al. (2006) use the Kullback-Leibler measure of the difference between probability distributions, and Hall (2009) uses entropy. In principle, such algorithms could be applied to the languages simulated by PyILM, but there are some complications that make this difficult. Specifically, these algorithms, or any other similar ones, require strong assumptions about what counts as an “environment” for the purposes of complementary distribution.

Environments can be defined at any arbitrary level - which should be considered? For example, suppose sound A occurs in the environments {t_i, a_a, s_o}, and sound B occurs in the environments {z_u, d_u, u_u}. Are these sounds in complementary distribution? If we consider just the segmental level, then the answer could be yes: Sound B only occurs before /u/, and Sound A occurs elsewhere.

If we think about features instead of whole sounds, then the situation becomes more complex. There are thousands of possible feature combinations to consider, depending on the feature system in use. On one analysis, both sound A and sound B have the same distribution: they can occur between vowels and they both follow coronal obstruents. On a different analysis, Sound A occurs between low vowels and after voiceless obstruents, while Sound B occurs between high vowels, or perhaps round vowels, and it follows voiced
Figure 3.1: Change in inventory size for Simulation 1

obstruents.

In dealing with a natural language, a linguist can make use of general knowledge about sound patterns, information from elsewhere in the language or related dialects, and intuitions about what constitutes a “natural” pattern, in trying to determine allophonic variation. For instance, if Sound B is labial(ized) and A is not, then an analysis of “B occurs before /u/, A occurs elsewhere” would be natural, since /u/ is also a round vowel. On the other hand, if A is voiceless and B is voiced, it might make more sense to refer to the fact that they only occur next to obstruents that match in voicing.

In the simulated languages of PyILM an algorithm searching for complementary distribution would have an enormous search space of all possible feature combinations to consider, as well as the problem of determining whether it is the left or right hand side (or both) of an environment that is most relevant. Therefore, as a way of avoiding some of these complications, I will make use of a much weaker definition of phoneme and allophone, where phonemic status is achieved by a sound when it exists in at least one word where it does not vary with another sound. Allophones are sounds that only exist as variants.

Maintaining this conceptual distinction between phonemes and allophones is very useful for interpreting simulation results, in particular when it comes to questions of inventory size. Counts of inventory sizes of natural languages tend to be counts of phonemic inventories, so it is useful to do this in PyILM as well. Figure 3.1 shows change inventory size for this simulation. The dotted line shows the core (phoneme) inventory, and the solid line show the total inventory (phonemes and allophones).

The figure shows that the size of the total inventory rises immediately, since [p] appears through misperception in the first generation. However, it is not yet a member of the core inventory, since it appears only as a variant of /b/. In generation 3, the size of the phoneme inventory rises as [p] has fully overtaken [b] somewhere in the lexicon. It now occurs in at
least one word where it does not vary with [b], though there are still many words where it remains a variant.

Immediately in the next generation, the phoneme inventory size drops again. Looking back at Table 3.2, this is because there has been a complete reversal in the language, and [b] has now become an allophone of /p/, that is, [b] only exists in words where it varies with [p]. This persists until Generation 5, and then [b] disappears completely. The consonant inventory for the remainder of the simulation is /p,q,n/.

The reason that /b/ disappears entirely, rather than continuing to co-exist with /p/, is that the phonotactics are restricted to VC syllables for the purposes of this simple illustration. If the language allowed onsets, then any onset [b] would remain a /b/ forever, since no misperceptions target that environment. The length of time that a language spends in the “doublet” stage of having alternative pronunciations depends on the frequency with which misperceptions occur, and the frequency of the words containing segments subject to misperception, which are parameters that will be analyzed in more detail throughout this chapter.

This transition from /b/ to /p/ in PyILM, or any other change like it, is a simplification of the real-world phenomenon of phonemicization, where phonetic effects eventually result in the appearance of a new phoneme in the inventory. Bermúdez-Otero (2007) describes four phases to this process. In the first phase, a new sound is introduced through “some physical or physiological phenomenon” (Bermúdez-Otero 2007, p. 7), and the language gains a phonetic variant of an existing sound. In the second phase, this variation becomes more categorical and what was once mostly a phonetic effect becomes a conditioned phonological alternation. The third phase is called re-analysis, where the domain of application for a phonological rule starts to change. It may become conditioned to a morphological environment, and lexical exceptions may appear. In the final phase, the original phonetic conditions become opaque, and the sounds become lexicalized, or the phonological rule becomes a morphological one. PyILM does not simulate all four phases, but there are clear parallels: a sound emerges in one context through misperception, varies with another sound for a period of time, then finally lexicalizes (since there is no morphology in PyILM).

### 3.3 Simulation 2 - A single gradual change

In the example above, sound change occurred when a learner misperceived certain tokens of a devoiced /b/ to be different enough from the “normally” voiced /b/, that a new category was assigned to these tokens. This new category existed alongside /b/ for a short period, then eventually dominated the lexicon, replacing /b/ in every instance. This is representative of scenarios in natural language where a sound first enters a language as an allophone, then becomes a phoneme. It is not necessary that an allophone completely replace a phoneme in a simulation, but the phonotactics of Simulation 1 were so restrictive that there was no other possible outcome. With more complex phonotactics, both /b/ and
/p/ would have been in the language at the end of the simulation.

The appearance of /p/ or the disappearance of /b/ was abrupt, occurring suddenly in some lexical items at some generations. It is also possible for a sound to change more gradually, by lowering the salience of a misperception, but increasing its frequency. This will have the effect of slowly pushing category boundaries in a particular direction, rather than generating a new category at any point. Eventually, the value of this phonetic dimension for a particular sound category will be considerably different from when the simulation started, and the phonetic properties of a sound category will have shifted far enough that the feature values will have flipped.

For this simulation, the same configuration file was used as in the previous section, but with two small changes. The simulation ran for 20 generations, instead of 10. There was a change to the misperception so that the devoicing misperception is twice as likely to occur, but its effect is only half as strong.

The same end-state inventory is achieved in both Simulation 1 and Simulation 2: the language has /p/ but not /b/. The main difference is that inventory size never changes in Simulation 2. The sound that is originally a /b/ has a voicing value that drops slowly over time. In generation 2, it has fallen enough to be classed as a [−voice] sound, but it straddles a perceptual boundary so in generation 3, it bounces back up slightly to the [+voice] side. In generation 4 it drops down to [−voice], where it stays for the remainder of the simulation.

### 3.4 Misperceptions and phonetic similarity

Having a high misperception salience means that learners are more likely to assign tokens affected by misperception to a different segmental category than those not affected by misperception. If this is combined with a high probability of misperceptions occurring, then the inventory will undergo more abrupt, categorical changes, as in Simulation 1. Lower salience values combined with higher probabilities of misperceptions leads to gradual sound change, as in Simulation 2.
Misperception salience interacts with another parameter, called `minimum_activation_level` (see section 2.2.2.14). This parameter is used during the learning phase, and it acts like a threshold for phonetic similarity in sound categorization. It controls how similar a token must be to a given category in order for the learner to consider including that token in that category. If a learner hears a sound that fails to meet this threshold, then the sound will be assigned to a new category. This parameter must have a value between 0 and 1. Setting it all the way to 1 means that input sounds must match existing categories exactly. This tends to lead to a proliferation of segment categories, since it is quite rare for exemplar tokens to be exact matches. Setting it to 0 means that nothing is ever too dissimilar, and all input tokens after the first will count as exemplars of whatever the first was categorized as.

These extreme values lead to unusual results, with segment inventories that look nothing like those of natural languages. More “normal” looking inventories emerge with values in the range of .5 to .7. Some results with different values are shown in Figure 3.2. Each of these simulations was run with the same configuration file.

![Change in inventory size over multiple simulations](image)

Figure 3.2: Results for various values of `minimum_activation_level`.

Figure 3.3 illustrates the interaction between misperception salience and `minimum_activation_level`. The figure shows the results of using a different value of `minimum_activation_level`, with each plot displaying change in inventory size for five different simulation runs, all using the same initial conditions, varying only the salience of misperceptions.
Figure 3.3: Varying misperception salience across three different values for \texttt{minimum\_activation\_level}. Misperception salience is shown in the legend. Simulation (a) uses a value of 0.2, Simulation (b) uses a value of 0.5 and Simulation (c) uses a value of 1.0

When the \texttt{minimum\_activation\_level} parameter is very low, as in Simulation (a), the salience of misperceptions hardly matters. The learning algorithm collapses all the segments into a single category. Even highly salient misperceptions cannot create segment categories with enough perceptual difference from the one existing category.

In Simulation (b), the minimum activation level is .5, so there is greater potential for misperception to create new categories. Growth in inventory size can be used as an indicator for when this occurs. Lower salience values produced smaller inventories while greater salience led to the creation of new categories quite quickly. Finally in Simulation (c), the high salience of misperceptions only speeds up growth in inventory size.

3.5 Simulation 3 - Interactions between sound changes

In the previous simulations, there was only a single sound change that could occur. This example gives a slightly more complex simulation in which sound changes can interact with each other. Simulation 3 uses the same configuration file as Simulation 1: the initial inventory is /b, q, n/ and the phonotactics are set to VC. The only difference is that there are now two misperceptions:

\textbf{Devoicing} [+voice, −son, −cont] segments have their [voice] value reduced by \( .5 \) in the environment of \( \# \) (\( p = .25 \))

\textbf{Lenition} [−voice, −son, −cont] segments have their [cont] value increased by \( .5 \) in the environment of \( \# \) (\( p = .25 \))

The first is the same final devoicing change used in Simulation 1. The second is a lenition process where voiceless stops become fricatives, also in final position. This means it is possible for /q/ to be affected by Lenition from the initial generation. On the other hand, /b/ is not affected, since it is voiced.
Once final devoicing has had an effect on /b/, however, the resultant [p] sound will be available for misperception as a labial fricative, perhaps /f/, creating a feeding relationship between the changes. Figure 3.4 shows some of the inventories that appeared over the course of Simulation 3.

<table>
<thead>
<tr>
<th>Generation 0</th>
<th>Generation 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Labial</strong></td>
<td><strong>Labial</strong></td>
</tr>
<tr>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>/b/</td>
<td>/b (p)/</td>
</tr>
<tr>
<td>Nasal</td>
<td>Nasal</td>
</tr>
<tr>
<td>/n/</td>
<td>/n/</td>
</tr>
<tr>
<td>Fricative</td>
<td>Fricative</td>
</tr>
<tr>
<td>/q/</td>
<td>/x/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 2</th>
<th>Generation 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Labial</strong></td>
<td><strong>Labial</strong></td>
</tr>
<tr>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>/b (p)/</td>
<td>/b (p)/</td>
</tr>
<tr>
<td>Nasal</td>
<td>Nasal</td>
</tr>
<tr>
<td>/n/</td>
<td>/n/</td>
</tr>
<tr>
<td>Fricative</td>
<td>Fricative</td>
</tr>
<tr>
<td>(/)/</td>
<td>/x/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 10</th>
<th>Generation 15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Labial</strong></td>
<td><strong>Labial</strong></td>
</tr>
<tr>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>/p/</td>
<td>/p/</td>
</tr>
<tr>
<td>Nasal</td>
<td>Nasal</td>
</tr>
<tr>
<td>/n/</td>
<td>/n/</td>
</tr>
<tr>
<td>Fricative</td>
<td>Fricative</td>
</tr>
<tr>
<td>(/)/</td>
<td>/x/</td>
</tr>
</tbody>
</table>

Table 3.4: Comparisons of several generations in Simulation 3

As the output shows, in generation 1 some changes have already happened. The /b/ has devoiced to [p] on some occasions, adding [p] as a new allophone of /b/. The /q/, which is already voiceless, has also been affected by the lenition misperception, and has also gained an allophone, and adds the first fricative to the inventory.

In generation 2, some instances of [p] have lenited to [φ], which is now actually counted as a second possible variant of /b/. PyILM will not consider it to be an allophone of /p/, since /p/ has no independent existence yet in the simulation. Another way of thinking about this is that the language has only a single labial sound at this point, with three possible pronunciations. By generation 6, /p/ has become the phoneme for the labial set: it has completely replaced /b/ in a certain number of words, and now [b] only ever appears as a variant of /p/. At generation 10, the voiced labial has completely disappeared, and both /p/ and /φ/ are considered phonemes. The uvular stop has also disappeared at this point, replaced in every instance by a fricative. By generation 15, the language is back to an inventory of the original size, with two fricatives replacing the two original stops. The nasal, meanwhile, has remained completely unaffected the entire time.
Figure 3.4 depicts how the total and core inventories are changing. Between generation 0 and 10, there is a large degree of allophonic variation. This settles down at generation 10 as [b] and [q] disappear as possible variants. The total inventory then drops once more when /p/ is eventually replaced by /φ/.

![Graph showing change in inventory size for Simulation 3](image)

**Figure 3.4: Change in inventory size for Simulation 3**

Setting up a feeding or bleeding relationship such as this can be quite difficult without careful manipulation of parameters. In this case, there is sure to be feeding that will happen because the initial /b/ is not subject to the lenition, only /p/ is. And /p/ is, by design, the outcome of the other misperception. If this simulation had been run with a random selection of segments, or with more complex phonotactics, there would be no guarantee this would occur. One misperception might never be triggered because the particular segment type or context is lacking in the language.

This simulation also illustrates why it is difficult to use PyILM to simulate the evolution of any specific natural language. For instance, suppose a simulation was seeded with a lexicon of Old English, and the set of misperceptions was configured to include all known sound changes from Old English to Modern English. There is no guarantee that all of the misperceptions would occur in the same order in a PyILM simulation as they did in the real world. So long as the conditions for a misperception are met in the lexicon, it is possible for a sound change to occur. It is not possible to force a particular ordering, at least not without introducing an unnecessary amount of teleology.
### Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>b,d,t,k,l,z,m,n,i,e</td>
</tr>
<tr>
<td>Lexicon size</td>
<td>30</td>
</tr>
<tr>
<td>Generations</td>
<td>20</td>
</tr>
<tr>
<td>Phonotactics</td>
<td>CVC</td>
</tr>
<tr>
<td>Invention rate</td>
<td>0</td>
</tr>
<tr>
<td>Min word length</td>
<td>1</td>
</tr>
<tr>
<td>Max word length</td>
<td>5</td>
</tr>
</tbody>
</table>

### Misperceptions

<table>
<thead>
<tr>
<th>Misperception</th>
<th>Target</th>
<th>Feature</th>
<th>Salience</th>
<th>Environment</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final devoicing</td>
<td>+voice,-son</td>
<td>voice</td>
<td>-0.5</td>
<td>#</td>
<td>0.025</td>
</tr>
<tr>
<td>Final lenition</td>
<td>-voice,-son</td>
<td>cont</td>
<td>0.5</td>
<td>#</td>
<td>0.025</td>
</tr>
</tbody>
</table>

Table 3.5: Configuration for Simulation 4

### 3.6 Simulation 4 - CVC language

This simulation uses the configuration details shown in Table 3.5. For this simulation, the phonotactics are slightly more complex, allowing CVC, CV, V, and VC syllables. Words are still capped at a single syllable, however. The initial inventory is specified in a file as /b, d, t, k, f, z, m, n, a, i, e/. The misperception file is the same as Simulation 2, that is, there is a chance of final devoicing and of final lenition of voiceless stops.

We know what is in the initial inventory, so we can make a few guesses about how the simulation will turn out, given enough time. Misperceptions only target final position, so if the lexicon contains at least one word with each segment in initial position, then all of the initial phonemes are guaranteed to survive until the end of the simulation. As it stands, the vowels and the nasals are sure to survive anyway, so long as they appear in at least one word, since no misperceptions target them at all.

Assume that in fact all the segments appear in initial position, so the inventory will not shrink over time. How can the inventory grow? Consider first the labial set. The segment /b/ will permit the creation of /p/ through devoicing. This /p/ is transient, though, and inevitably all examples of it will lenite, just as in the last simulation. No /p/ can “survive” because all instances of this category are found in final position, which is exactly where lenition applies. This extremely restricted distribution is due to the fact that there was no /p/ in the initial inventory; /p/ appears through devoicing which occurs only in final position. If /p/ had been part of the initial inventory, then it might have occurred in initial position, which would have protected it from the lenition change.

Once /p/ has lenited to a fricative, there are no more changes that can take place. The final labial inventory will include /b, p*, f/ where /p*/ is either a labial stop or a labial fricative, depending on how long the simulation has been running. This fricative may or may not merge with the original /f/. A merger will take place only if /p*/ and /f/ are identical on every dimension except [continuant]. This also implies that /b/ and /f/ are identical on every dimension except [voice] and [continuant], since /p/ descends from /b/. If, for any reason, /b/ and /f/ differ on some other feature, then when /p/ lenites, the resulting category will be considered a different category from /f/.

77
For example, in the feature system of Hayes (2011), [labiodental] is feature. Since the original /b/ is \([-\text{labiodental}\)] in this system\(^1\), then /p*/ will be \([-\text{labiodental}\)] because the Lenition misperception only affects the feature [continuant]. In this case, /p*/ will differ from /f/ by both [continuant] and [labiodental], and will therefore be categorized as something other than /f/.

The coronals will have a somewhat different evolution. There is /z/ but not /s/, so it is expected that /s/ will eventually appear through misperception. In fact some kind of voiceless coronal fricative is almost certain to appear because the original /d/ should devoice to /t/, which is subject to the lenition misperception. It is possible that the voiceless coronal fricative that ultimately descends from /z/ will merge with the one that descends from /t/, but it will depend on specifically how the exemplar token values are distributed in a given simulation, and this process is partly random. The coronal inventory could eventually grow to /d, t, z, s\(_1\), s\(_2\)/, where the two /s/ segments represent potentially different descendants of original /t/ and original /z/.

The single dorsal /k/, if it only appears in final position, is doomed. It is certain to undergo lenition, and there are no original voiced dorsals that could create a “replacement” /k/ through devoicing. The final dorsal inventory is therefore going to be /x, k/, if /k/ appears in initial position, otherwise it will simply be /\(x/\).

Again, these predictions depend on some assumptions about the initial lexicon of the language, and whether or not the relevant segments all appear in the relevant environments. Running this same simulation multiple times with different random seeds will produce different outcomes. The plot in Figure 3.5 shows the change in (total) inventory size over five simulations using the same initial conditions, but with different random seeds. In the simulations with larger final inventories, sounds appear in a greater variety of environments, increasing the probability that they survive the entire simulation, since they are more likely to appear in environments not targeted by a misperception. In the simulations with smaller final inventories, there was less diversity in sound distributions, and some sounds disappeared because they occurred only in environments targeted by misperception.

Note that not all simulations started with exactly the same number of sounds: some started with 10 and some with 11. All languages were given the same configuration file with the same inventory, but on some random seeds, not all of these sounds were actually sampled during the construction of the initial lexicon. There is a simulation parameter called auto_increase_lexicon_size (see section 2.2.2.15) which would force every simulation to use all 11 sounds, but it was set to False for these cases.

Table 3.6 gives some snapshots of a language actually generated by one of these random seeds. The labials turned out as predicted. The phoneme /b/ first develops an allophone [p], which then becomes a phoneme, and which then lenites and becomes a fricative. In this case, it did not merge with the existing /f/, and there are two labial fricatives in the

---
\(^1\)In Hayes' system, [labiodental] is actually a unary feature, so /b/ would simply not have this feature at all. However, since features in PyILM cannot be unary, /b/ would be considered [-labiodental].
final inventory. Inspecting the simulation, it appears that /f/ is [-distributed] while /θ/ is [+distributed], which is a feature it inherits from the original /p/.

The coronals evolved more or less as expected as well. The phoneme /d/ devoiced to [t], which gave rise to [θ], which eventually achieved phonemic status. The phoneme /t/ rises and falls throughout the simulation, as some tokens of [d] devoice, then lenite. Some instances of /d/ still remain in final position, so they allow for new devoicing which leads to new lenitions (that all merged with the first /θ/). If the simulation were run long enough, eventually /t/ would completely overtake /d/ in all final positions, and then eventually lenite to /θ/.

The dorsal stop was lost, and replaced by a fricative, which was predicted. However, this actually did not happen entirely due to misperception. This segment had a curious evolution. The original /k/ appeared in three words: /ki/, /ik/ and /ak/. It early on acquired an allophone [x] in final position. This [x] became an increasingly common variant until it was the dominant pronunciation in two out of three words: /ik/ > /ix/ and /ak/ > /ax/. Then the learner in generation 10 decided to group /x/ and /k/ into a single category. Even the /k/ in initial position, not affected by misperception, merged with /x/. Why did this occur?

Inspecting the simulation more closely revealed that the initial /k/ category had been seeded with exemplars that happened to have extremely low values on the [continuant] dimension, so that most tokens produced had a value of 0.1 or less. The final lenition misperception boosted production values by +.5, which created tokens that only barely passed the threshold for a learning agent to categorize something as [+continuant], so the new /x/ category had values that straddled a perceptual boundary.

At generation 10 it appears as though the learner failed to notice any significant differ-
ence between any /k/ or /x/ tokens produced by the previous generation, and categorized them all as [continuant], that is, /k/ became the new phoneme. This created a category with a large degree of variation in [continuant] values. Misperception continues to act on transmission to the next generation, which pushed average [continuant] token values higher. The learner at generation 11 also only learned a single velar category, but this time [+continuant], that is, /x/ became the phoneme. Since there is no misperception that makes word-initial tokens any less continual, there is no way for /k/ to return to the inventory, and this collapse of categories is essentially permanent.

<table>
<thead>
<tr>
<th>Generation 0</th>
<th>Generation 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop</strong></td>
<td><strong>Stop</strong></td>
</tr>
<tr>
<td>b</td>
<td>b (p)</td>
</tr>
<tr>
<td>d</td>
<td>d (l)</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td><strong>Nasal</strong></td>
<td><strong>Nasal</strong></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td><strong>Fricative</strong></td>
<td><strong>Fricative</strong></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>z</td>
<td>z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 6</th>
<th>Generation 11</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop</strong></td>
<td><strong>Stop</strong></td>
</tr>
<tr>
<td>b (p)</td>
<td>b (p)</td>
</tr>
<tr>
<td>d (t)</td>
<td>d (l)</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td><strong>Nasal</strong></td>
<td><strong>Nasal</strong></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td><strong>Fricative</strong></td>
<td><strong>Fricative</strong></td>
</tr>
<tr>
<td>f</td>
<td>f (θ)</td>
</tr>
<tr>
<td>z</td>
<td>z (θ)</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation 12</th>
<th>Generation 19</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop</strong></td>
<td><strong>Stop</strong></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>d (l)</td>
</tr>
<tr>
<td><strong>Nasal</strong></td>
<td><strong>Nasal</strong></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td><strong>Fricative</strong></td>
<td><strong>Fricative</strong></td>
</tr>
<tr>
<td>φ f</td>
<td>φ f</td>
</tr>
<tr>
<td>z θ x</td>
<td>z θ x</td>
</tr>
</tbody>
</table>

Table 3.6: Comparison of several generation in Simulation 4

### 3.7 Simulation 5 - Invention and the spread of new segments

In the previous examples, the new segments that are created by misperceptions are in competition with existing segments, and only one of them can win. Inevitably, it will be the one that is “preferred” by the misperception. These newly created segments, however, are more like replacements for the older segments, rather than truly new additions to the language. They never leave their original environments, because the invention rate has been set to 0.0 for the previous simulations. In this next example, the invention rate is raised to demonstrate how this affects the evolution of an inventory. The configuration details are shown in Table 3.7.

Agents inventing new words will draw from the total inventory of sounds, not just from the phoneme inventory. This makes it possible for allophones to become phonemes, because they can appear in an invented word in an environment where they are not in variation
with another sound. This is analogous to a process that is known to happen in natural language where words are borrowed containing an allophone in a novel environment, which can lead to that allophone taking on phonemic status. For example, in Old English [f] and [v] were allophones of a single phoneme, with [v] occurring intervocally and [f] occurring elsewhere. Over time, English borrowed French words that contained a [v] in positions other than between vowels (McMahon 2002). This created overlapping distributions of [f] and [v], which resulting in [v] eventually taking on phonemic status.

The misperceptions are the same as the previous simulations: a 25% chance of word-final devoicing and word-final lenition. The combination of misperceptions and inventions creates different outcomes than the previous simulations without invention. For instance, consider just the coronals. In the initial inventory there is a voiced coronal stop /d/, but no voiceless counterpart. The voiced one appears in both word-initial and word-final position in the initial lexicon. After several generations of the simulation, all of the /d/ in final position have devoiced, and there is now a voiceless coronal stop /t/ in the inventory. This newly created stop is now subject to final lenition, and eventually all instances of it become voiceless fricatives, returning the language to a state of only having the one (voiced) coronal stop.

Table 3.7: Configuration for Simulation 5
This voiced stop will then have a restricted distribution - it will only be found in word-initial position, because no misperceptions operate there. In previous simulations, no more change would be possible at this point, since misperceptions can have no more effects. However, in this simulation the invention rate is greater than 0, so there is the possibility that an agent can create new words and put the voiced stop back into final position. This makes it now a target of final devoicing, and a voiceless stop will eventually re-join the inventory. It is also possible that during the period of time where /t/ exists as a phoneme, an agent will invent a new word that contains a /t/ in initial position, shielding at least some instances of /t/ from lenition, making it a more permanent member of the inventory.

3.8 Summary

In this chapter, I demonstrated how inventories evolve in PyILM, and how various simulation parameters can affect this evolution. The notion of phonemes and allophones in PyILM were introduced, as they differ somewhat from the common use of these terms in phonological theory. A sound is considered to be a phoneme in PyILM if it occurs in at least one word in the lexicon where it does not vary with another sound. A sound is considered to be an allophone if it only ever occurs as a variant of other sounds. There were five simulations presented in this chapter.

Simulation 1 showed how inventories can change through the abrupt introduction of a new sound, and Simulation 2 showed how categories can shift slowly over time. The different outcomes depended on the values of different simulation parameters. When misperceptions have a high salience, this tends to lead to the emergence of allophones. For instance, suppose a simulation has an intervocalic lenition misperception with a high salience, and
a lexicon has /b/ between vowels. A word such as /aba/ will quickly obtain two possible pronunciations: [aba] and [ava]. Initially, the [v] sound will be a variant of /b/, but some number of generations, the word will come to be pronounced uniquely as [ava] and /v/ will enter the inventory as a phoneme.

When misperceptions have a lower salience, sounds in an inventory tend to gradually change categories, without the appearance of an intermediate phoneme. For example, suppose a simulation has a low-salience intervocalic lenition misperception. A word like /aba/ will continue to be pronounced as [aba] for a few generations, but the effect of the misperception will slowly drag the [continuant] values of the /b/ segment (in this word) higher. Eventually, some learner will acquire the word as /ava/, and it will have a unique [ava] pronunciation. In contrast to the high-salience simulation, it is less likely that a situation will arise where both [aba] and [ava] are possible pronunciations in the low-salience simulation.

There is also an interaction between the misperception salience the minimum_activation_level parameter. When this parameter is set very low (close to 0) then all segments in a simulation will tend to collapse into a single category. If the parameter is set very high (close to 1) then there is an extreme proliferation of segment categories. These effects are very strong, and will occur regardless of the salience and frequency of any misperceptions.

Simulation 3 increased the number of misperceptions and included some feeding relationships, for example a lenition process that only affect voiceless sounds which were themselves the product of a devoicing misperceptions.

Simulation 4 demonstrated how phonotactics can influence the outcome. This is due to the context-sensitive nature of sound changes. A language with only CV syllables has exactly two contexts for consonants: word-initial or intervocalic (assuming a word of at least two syllables). This limits the number and type of misperceptions that could potentially apply. On the other hand, a languages with CVCC syllables has a greater variety of environments in its lexicon, which means that a greater variety of sound changes could potentially take place. The issue of phonotactics will be discussed in much more detail in the Chapter 5.

Simulation 5 introduced the concept of inventions. Invention has two major effects on the outcome of a simulation. One is that invention creates new words with new environments, allowing misperceptions to apply to sounds that might not apply in other words. The second possibility is that allophones can be selected by the invention algorithm and places into new contexts where they do not vary with any other sounds, instantly achieving the status of phonemes.

This builds up the basic foundations of simulations in PyILM. Now more complex simulations can be considered, with the aim of trying to model the evolution of natural language inventories. In Chapter 4, I will review the typology of natural language inventories, before returning again to PyILM in Chapter 5, with the aim of simulating these typological facts.
Chapter 4

Natural language consonant inventories

4.1 Inventory size

4.1.1 Overview

Sound inventories are extremely diverse. One of the most obvious ways in which they differ is in the number of sounds they contain. Counts of inventory size depend partly on what is being counted. It is common in linguistics to make the distinction between the “surface” or “phonetic” inventory of a language, which consists of the sounds that are physically articulated, and the “underlying” or “phonemic” inventory, which consists of abstract mental categories assumed to be acquired by a learner of a language.

Collecting a complete phonetic inventory, a set of all the speech sounds in a language, is actually not feasible, since no two speech productions are exactly alike, and this collection would be infinite in size. Speech sounds are instead grouped into a finite set of categories, with categorization typically done through the use of articulatory or acoustic features. The International Phonetic Alphabet, for example, is a very widely used system for categorizing speech sounds based on articulation. Major category features for consonants in the IPA include place of articulation, manner of articulation, voicing, and airstream mechanism.

The phoneme inventory of the language is based on an analysis of the lexicon. The distribution of a sound in the lexicon determines its phonemic status. Phonemes are usually argued for on the basis of contrast, with minimal pairs being the best evidence. Sounds that never appear in the same environment, i.e. have complementary distribution, are considered allophones of a single phoneme. There is sometimes an additional requirement that allophones bear some phonetic resemblance to each other. For instance, in English the sounds [h] and [u] are in complementary distribution, with [h] appearing only in syllable initial position, and [u] appearing in non-initial position. Despite this, the two sounds are not analyzed as allophones of a single phoneme because they are phonetically quite different.
Phonetic description of 19 surface consonants

<table>
<thead>
<tr>
<th></th>
<th>Labial</th>
<th>Dental</th>
<th>Dento-</th>
<th>Alveolar</th>
<th>Velar</th>
<th>Glottal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>[b]</td>
<td>[p]</td>
<td>[d]</td>
<td>[t]</td>
<td>[g]</td>
<td>[k]</td>
</tr>
<tr>
<td>Fric</td>
<td>[ð]</td>
<td>[θ]</td>
<td>[s]</td>
<td></td>
<td>[ŋ]</td>
<td></td>
</tr>
<tr>
<td>Nas</td>
<td>[m]</td>
<td>[n]</td>
<td></td>
<td></td>
<td>[l]</td>
<td></td>
</tr>
<tr>
<td>Lat</td>
<td>[l]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tap</td>
<td>[r]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trill</td>
<td></td>
<td></td>
<td>[ɾ]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glides</td>
<td>[w]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Phonetic description of 11 contrastive consonants

<table>
<thead>
<tr>
<th></th>
<th>Bilabial</th>
<th>Dental</th>
<th>Dento-</th>
<th>Alveolar</th>
<th>Velar</th>
<th>Glottal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>[b]~[p]</td>
<td>[t]~[tʰ]</td>
<td></td>
<td></td>
<td>[k]<del>[kʰ]</del>[g]</td>
<td>[ʔ]</td>
</tr>
<tr>
<td>Fric-stop</td>
<td>[ð]<del>[θ]</del>[s]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fric</td>
<td></td>
<td></td>
<td>[s]</td>
<td></td>
<td>[ŋ]~[n]</td>
<td></td>
</tr>
<tr>
<td>Nas</td>
<td>[m]</td>
<td></td>
<td></td>
<td></td>
<td>[l]</td>
<td></td>
</tr>
<tr>
<td>Lat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhot</td>
<td>[ɾ]~[ɾ]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glides</td>
<td>[w]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1: The inventories of Palauan, from Morén-Duolljá (2005)

The inventory of Palauan is a good example of how some of these decisions can affect what gets counted in an inventory. The tables combined in Figure 4.1 come from Morén-Duolljá (2005). The top table gives the approximate phonetic inventory, which is something like the set of all articulatorily distinct sounds found in Palauan speech. The bottom table gives what Morén-Duolljá calls the “contrastive” consonants. Each box is a phonemic category and the symbol ~ is used to indicate the multiple possible pronunciations for a sound in that category.

There is a difference of 8 sounds between the two tables. There are three kinds of velar stops that are articulated in Palauan - voiceless unaspirated, voiceless aspirated, and voiced - but they are all considered variants of a single velar phoneme. There are two reasons for grouping them together as allophones: (1) they are phonetically similar, (2) they appear in complementary distribution in the lexicon. Specifically, [kʰ] occurs in final position, [g] appears between vowels, and [k] appears elsewhere. Figure 4.2 provides a word list and summary of this distribution.

Since [k] is the least predictable of the allophones, it is also assumed to be the underlying phoneme. In constructing a phonemic inventory of Palauan, the velar stop category would be represented using the symbol /k/. The aspirated and voiced velars would not be represented.

85
This can be compared to another, rather more simple case, which is the bilabial nasal. The sound [m], according to Morén-Duolljá (2005) is found in a variety of environments, and there are no noticeable variations in pronunciation. There is one other nasal in the language, which appears nearly everywhere that [m] does, so there is no complementary distribution that might suggest an allophonic relationship. Paluan is therefore assumed to include an underlying category /m/ which would appear in a phonemic inventory.

The focus of this chapter will be phoneme inventories. One major reason for this is that there exist several large databases of information about phoneme inventories. Additionally, the abstract categorical nature of phoneme inventories makes them somewhat easier to collect and analyze, compared to the more gradient nature of phonetic data. Major databases that will be frequently referenced in this chapter are UPSID, P-base, and WALS, which are described below.

UPSID is the UCLA Phonological Segment Inventory Database. UPSID was the first major database of inventories, and is extremely widely used. It was first published as Maddieson (1984) with 317 languages. In Maddieson and Precoda (1989), it was expanded to 451 inventories. The database attempts to be genetically balanced, to represent an even spread of the world’s languages. UPSID has a a very simple web interface at: http://web.phonetik.uni-frankfurt.de/upsid.html.

P-base was created as part of Jeff Mielke’s dissertation work (Mielke 2008). It contains the inventories of 628 varieties of 548 spoken languages. The languages in the database are those that Mielke could find in grammars available at the Ohio State University and Michigan State University libraries (Library of Congress PA-PM). In addition to the inventories, P-base also includes any information about the patterning of sounds that was available in the grammars. P-base has a graphical user interface with functions for finding natural
classes, calculating feature economy, and comparing inventories. It can be downloaded at
http://pbase.phon.chass.ncsu.edu/

WALS is the World Atlas of Language Structures (Dryer and Haspelmath 2013), and
contains information from more than 1,000 languages. WALS is not limited to phonological
inventories, unlike the previous two resources, but rather contains information about nu-
merous aspects of language, including morphological and syntactic information. It is also
not a single database, rather it is a collection of individual chapters written by different
authors, and each chapter may sample a different set of languages. An interesting feature
of WALS is the ability to display a map of the world, with individual languages tagged
and colour-coded for particular features. The information available in WALS comes from
a variety of sources, and each language has its sources listed. It is not possible to look at
a specific phoneme inventory of a language in WALS. Instead, the information is packaged
in a more coarse-grained way, by grouping languages into categories. For example, Feature
6A in WALS (Maddieson 2013b) is titled “uvular consonants” which categorizes languages
into four categories: those with uvular stops, those with uvular continuants, those with
both, and those with neither. This makes it more useful for broad, typological studies,
and somewhat less useful for the study of individual languages. It is available online at
www.wals.info.

Large databases like these are created from a diverse array of sources, and constructed
with different goals in mind, so it is inevitable that there will be disagreements. One
example of this is the way that Jacaltec (Mayan, Mexico) is described in UPSID and P-
base. In UPSID the stop series for this languages is listed as three voiceless aspirated stops
/pʰ, tʰ, kʰ/, two ejective stops /t’,k’/, and two implosives /b<, q</ (the < symbol marks
implosives in UPSID). In P-base, the same language is listed with three unaspirated stops
/p, t, k/ and there is a set of voiced stops /b, d, g/ labeled “marginal”. No implosives are
listed at all. The uvular is included in the ejective series /t’, k’/, q’/. The symbol /b’/
also appears, which represents a glottalized /b/, but this is apparently distinct from the
implosive (which is not listed for Jacaltec, but does appear for other languages in P-base).
The two databases even give similar sources: P-base cites (Day 1973), which appears nearly
identical to the (Day 1972) reference given in UPSID. Perhaps P-base simply replaced the
UPSID symbol < with the apostrophe, although UPSID additionally cites (Craig 1977),
which could be another source of the disparity.

In UPSID, the median inventory size is between 28 and 29, meaning that half of the
inventories have 29 or more sounds, while the other half have 28 or fewer. A majority of
languages (70%) have between 20 and 37 segments.

The smallest known inventory is that of Central Rotokas. The inventory is described in
Firchow and Firchow (1969), who divide it into two classes of sounds: voiceless and voiced.
The voiceless category consists of three voiceless stops (labial, coronal, and velar). The
voiced category consists of three sounds that Firchow and Firchow transcribe as a voiced
bilabial fricative, a voiced coronal tap, and a voiced velar stop. The phonemic inventory is
shown in Figure 4.3.
However, they note there is considerable amount of free variation in the realization of the voiced series. The bilabial is variously realized as a fricative, as a nasal, or even a full plosive. The coronal may surface as as a nasal, a lateral, a tap, or a plosive. And the velar can be a fricative, a nasal, or a plosive. The voiceless coronal has slightly more conditioned variation, and appears as [s] or [ts] before [i] but [t] elsewhere. This means that voicing and place of articulation are contrastive in Central Rotokas, but manner of articulation is not.

More recent field work by Robinson (2006) suggests that nasals actually are distinct in the Aita dialect of Rotokas. Robinson reports minimal pairs for all three places of articulation: buta “time” vs. muta “taste/feel”, dito “hole” vs. nito “remove embers”, and kati “burn” vs. yati “bend”. Other phonological alternations are similar between the dialects: the non-nasal voiced bilabial of Aita alternates between a stop and a fricative, the coronal alternates between a stop and a tap, and the voiceless coronal surfaces as [s] before [i].

Robinson posits that Proto-Rotokas must have had nasal consonants, and they were lost in the Central dialect. He justifies this by it being the simpler analysis, suggesting a context-free historical change where all [+nasal] sounds in Proto-Rotokas became [−nasal] sounds in Central Rotokas. Robinson’s account actually relies on [+nasal] sounds only becoming *underlyingly* [−nasal], since nasals still exist as surface variants of voiced sounds in Central Rotokas. What has changed is their contrastive status.

According to Maddieson (2013a), there are two major regions of the world where small inventories tend to predominate (where “small” means 6-14 consonants). One of these regions is the Pacific Islands, where most of the languages belong to the Oceanic branch of the Austronesian family. The other region is the northern part of South America, with languages in Ecuador, Columbia, Venezuela and parts of Brazil. Some of these language belong to large genetic grouping, such as Arawakan or Carib, while others remain unclassified or are considered isolates.

Perhaps the best known of these South American languages is Pirahã (Everett 1986). Pirahã has 8 consonants, /p, t, k, ?, b, g, s, h/, which is 2 more than Central Rotokas. However, Pirahã has fewer vowels, so the overall phoneme count is the same as Rotokas.
### Allophonic Variation among the Languages

Allophonic variation also occurs within the languages, so the sounds have different surface realizations. For example, in Ubykh, the voiced stops in Pirahã have nasal variants, although curiously /g/ surfaces as [n], rather than the expected [ŋ]. The fricative /s/ may surface as [ʃ] or [h], and before /i/ the sound /t/ becomes /tʃ/.

At the other extreme end of inventory sizes are the Kho-Eh languages, which tend to have extremely large consonant inventories. The !Xóó language (Kho-Eh; Botswana) is generally cited as having the largest consonant inventory, and the number given is typically more than 100 sounds, although authors disagree on exactly how many. P-base lists 117 consonants, of which 70 are clicks, based on Traill (1985). The non-click inventory also features an impressive number of coronal affricates and ejectives. On the other hand, UPSID lists only 94 consonants, of which 47 are clicks, based on Snyman (1969). Either way, the inventory is considerably larger than most others. !Xóó also exhibits a remarkably rich vowel inventory. There are vowel contrasts based on length, tone, pharyngealization, nasalization, and combinations thereof (though not all combinations are found).

### Outside the Kho-Eh Group

Outside of the Kho-Eh group, WALS Chapter 1 (Maddieson 2013a) shows a few regions around the world that tend to have large inventories, where “large” is defined as having 34 or more consonants. The Caucasus is one such region. It was here that the largest inventory outside of click languages was found, belonging to Ubykh, a Northwest Caucasian language that went extinct in the 1990s. According to the inventory in P-base, from Colarusso (1988), it had 81 consonants.

Of this total, 76 were obstruents; only 7 were sonorants. Forty-four of the obstruents were either fricatives or affricates, and the remainder of the inventory consisted of plosives. Like many other languages in this family, Ubykh had a very small phonemic vowel inventory, which P-base lists as consisting of only 2 vowels contrasting in height. Sec-

---

**Figure 4.4: The inventory of !Xóó, based on Traill (1985)**

<table>
<thead>
<tr>
<th>PULMONIC CONSONANTS</th>
<th>CLICK CONSONANTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vowels</strong></td>
<td><strong>Bilabial</strong></td>
</tr>
<tr>
<td><strong>High</strong></td>
<td><strong>Dental</strong></td>
</tr>
<tr>
<td>i, i, i</td>
<td>0</td>
</tr>
<tr>
<td>e, e, e</td>
<td>0</td>
</tr>
<tr>
<td>a, a, a</td>
<td>0</td>
</tr>
<tr>
<td><strong>Mid</strong></td>
<td><strong>Alveolar</strong></td>
</tr>
<tr>
<td>a, o, a</td>
<td>a</td>
</tr>
<tr>
<td>e, o, e</td>
<td>e</td>
</tr>
<tr>
<td>a, o, a</td>
<td>a</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td><strong>Palatal</strong></td>
</tr>
<tr>
<td>a, a, a, a, a, a, a</td>
<td>a, a, a, a, a, a</td>
</tr>
<tr>
<td>a, o, a</td>
<td>a, o, a, a, a, a</td>
</tr>
</tbody>
</table>

### P-BASE CONSONANTS

<table>
<thead>
<tr>
<th>STOP</th>
<th>FRICATIVE</th>
<th>NASAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, p</td>
<td>ts, ts', t's</td>
<td>m, n</td>
</tr>
<tr>
<td>D, d'</td>
<td>dz, dz'</td>
<td>n</td>
</tr>
<tr>
<td>T, t'</td>
<td>tk'</td>
<td>n</td>
</tr>
<tr>
<td>K, k'</td>
<td>glk'</td>
<td>n</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLICK CONSONANTS</th>
<th><strong>Lateral</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>f</td>
</tr>
<tr>
<td>Oh</td>
<td>hh</td>
</tr>
<tr>
<td>Q</td>
<td>ʔ</td>
</tr>
<tr>
<td>Qh</td>
<td>ʔh</td>
</tr>
<tr>
<td>Qq</td>
<td>ʔk</td>
</tr>
<tr>
<td>Ox</td>
<td>ʔx</td>
</tr>
</tbody>
</table>

---

---

---

---
ondary articulations are heavily used in Ubykh, with virtually every obstruent occurring as a plain version in addition to palatalized, labialized, ejective, and pharyngealized versions, and sometimes combinations (e.g. there was a labialized ejective alveolar affricate, and a palatalized labialized uvular ejective).

Another hotspot for large inventories, according to Maddieson (2013a), is the Pacific Northwest Coast of Canada and the United States. The larger language families found in this region are the Salish, Wakashan, and Athabaskan languages. These languages often have 40 or more consonants, and again there tends to be a greater use of secondary articulations, such as labialization, and glottalization (of both obstruents and sonorants) is also quite common.

### 4.1.2 Population size

Why are there different sizes of inventories? What prevents languages from all having, say, 25 consonants and 5 vowels each? This is due to the fact that inventories are unstable: languages gain and lose sounds over time, and each language follows its own unique path of changes, so it is very unlikely that all languages would end up with inventories of the same size. The more interesting question is what factors affect inventory size in the first place. Although one might expect phonology or phonetics to be relevant here, in recent years a more widely discussed factor was actually population size.

This issue received a great deal of attention when Hay and Bauer (2007) published an unexpected correlation: they found that there was a positive statistical relationship between inventory size and the size of the population of speakers of the language. Their corpus for analysis was the languages included in (Bauer 2007). Some of the languages were removed from consideration because it was not possible to get full vowel, consonant, and population data. Languages with no living speakers were also removed. Their final corpus included 216 languages. They additionally removed !Xôó and Accoli from certain analyses because of the size of their inventories (more than 4 standard deviations above the mean). They note that the corpus is not a random set, and is necessarily biased toward languages that have available data. This means a number of languages with very large populations (English, Hindi, Mandarin), but also many that are well-documented but with small populations (Diyari, Hixkaryana, Basque). The corpus is also not geographically balanced, and tended toward the Indo-European family and languages of the Pacific.

Figure 4.5 from Hay and Bauer (2007, p.13) shows the relationship between population size and inventory size in their corpus. They measured total inventory size (bottom right of the figure) as well as the relationship between population and sub-inventories, such as sonorants. In all cases, they found a significant correlation. Vowel inventories showed similar correlations between inventory size and population size, although Hay and Bauer report that vowel and consonant inventory sizes have no correlation with each other in this set (see also Maddieson (2007)). The authors additionally investigated the relationship between average inventory size and average population size for each family, and again found
Figure 4.5: Correlations between speaker population size (individual languages) and inventory size, from Hay and Bauer (2007).
a significant positive correlation. Figure 4.6, from Hay and Bauer 2007, p. 16, shows this relationship.

This correlation was later used in a highly controversial article by Atkinson (2011). He calculated that distance from Africa correlates with inventory size, such that languages spoken further away tend to have smaller inventories. He combines this with Hay and Bauer’s correlation to support a particular model of human migration out of Africa with successive “founder populations”: these populations are very small, which implies their languages would have small inventories. In his own words:

“If phoneme distinctions are more likely to be lost in small founder populations, then a succession of founder events during range expansion should progressively reduce phonemic diversity with increasing distance from the point of origin, paralleling the serial founder effect observed in population genetics.” (Atkinson 2011, p. 1)

This paper was heavily criticized, with responses largely focusing on the statistics. Cysouw et al. (2012) find several faults. Their main criticism is Atkinson’s choice of data. He used the “coarse-grained” summary of the UPSID database which is available in the World Atlas of Language Structures. Cysouw et al. tried to replicate Atkinson’s methods on the original UPSID data, and they report no significant correlation between population size and inventory size.
They also object to Atkinson’s use of the Bayesian information criterion (BIC) for determining geographical origin. Atkinson decided to allow locations which were as many as 4 BIC units away from the optimal origin, which Cysouw et al. view as an arbitrary, unjustified choice. The authors also tried to use the BIC method on the original UPSID data, and this time the model suggested an origin in either Africa or the Caucasus.

Donohue and Nichols (2011) used a different sample of 1,350 languages, rather than replicating Atkinson’s study. The found no significant correlation between population size and inventory size in their corpus. They did, however, find a small correlation when they looked at inventory size across geographical areas: inventories get somewhat smaller moving roughly west-to-east. Mean populations in the western regions (Africa, Europe) were larger and so were mean inventory sizes, whereas mean population size in the eastern regions (New Guinea, Oceania) were smaller and so were mean inventory sizes. However, when looking within a geographical area, or within language families, there was no significant trend. Relationships between population size and inventory size within several families are shown in Figure 4.7.

Donohue and Nichols (2011) also note that Atkinson’s model predicts a monotonic decrease in phonological inventory size as distance from Africa increases. In fact, around the world there are local “hotspots” for phonological complexity, that is, languages markedly more complex inventories than those nearby. For instance, within North America, the languages of the Pacific Northwest Coast are a hotspot. These are unexplainable in Atkinson’s model. Their conclusion is:

“A positive correlation between population size and size of phoneme inventory is critical to Atkinson’s argument, but such a correlation is not expected given current knowledge of socio-linguistics, typology, and historical linguistics, and it cannot be demonstrated cross-linguistically.” Donohue and Nichols (2011, p. 169)

On the other hand, Wichmann, Rama and Holman (2011) did find support for the correlation between population and inventory size. They used an extremely large corpus of 3,153 languages taken from the ASJP Database (Wichmann, Muller, Velupillai, Brown, Holman, Brown, Sauppe, Belyaev, Urban, Molochieva, Wett, Bakker, List, Egorov, Mailhammer, Beck and Geyer (2011)). The data consists of 40 concepts, and the corresponding words, in different languages. The 40 selected meanings come from the Swadesh list (Swadesh 1952) and most of the languages in their corpus had a word for 28 or more of these meanings. The database therefore does not provide direct access to phoneme inventories, only small lexicons. To estimate the inventory of a language, the authors took the available word list for a language and collected each symbol that appears in at least one word. In cases where more than one word was available for a given meaning, the authors took whichever word happened to be listed first in the database.

This method of counting leads to inaccuracies, and makes it unclear if this is a count of phonetic or phonological inventories. Wichmann, Rama and Holman nonetheless find
Figure 4.7: Relationship between population size (log scale) and inventory size for several language families, from Donohue and Nichols (2011)
that their numbers are comparable to UPSID, for those languages that appear in both databases: “word lists are approximately proportional to segment inventory sizes to a degree where it is meaningful to use SRs [Segments Represented in the word list] as proxies for segment inventory sizes when it comes to investigating correlations with other features, such as word length, populations size, and geographical distances”. Figure 4.8 shows the plotted correlations between population size and inventory size found in ASJP. Wichmann, Rama and Holman, p. 20 conclude that “we are able to confirm that...larger populations are associated with larger phoneme inventories, ... and that, finally, phoneme inventories diminish with distance from Africa.”

Moran et al. (2012) attempted to replicate Hay and Bauer (2007), this time using a larger set of 969 languages from the PHOIBLE database (Moran et al. (2014)). They too found a positive correlation, but an extremely weak one: the model predicts a increase of between 1.02 and 1.04 phonemes per tenfold increase in population size. Since their corpus includes languages with speaker populations ranging from 1 person to 840 million people, there is a predicted difference of only nine phonemes between the smallest and largest populations in the sample. This is shown in Figure 4.9 from Moran et al. (2012, p. 18).

Moran et al. conclude that the correlation, while present, is not big enough to be important.

“We believe that the magnitude of the relationship is not substantial enough to be of interest when viewed in light of the variation within and across genealogical groupings. Thus we find no compelling reason to consider population size as a potential causal factor in the development of phonological systems, and thus no
reason to postulate explanations, mechanisms, or reasons why such a pattern exists.” Moran et al. (2012, p. 896)

Although I agree with Moran et al., the fact remains that languages do differ in inventory size and no alternative explanations have been offered. The conclusion seems to have been that whatever determines inventory size, population size is not it. Is inventory size something that randomly fluctuates over time? Or could there be some factors that directly influence size?

Several years before Hay and Bauer (2007), the relationship between population and inventory size had already been discussed by Trudgill (2004), although his work is more of an argument for how such a relationship could develop, and does not include any statistical model. In particular, Trudgill’s paper looks at how contact between populations could be a factor that affects inventory size.

In the case of long-term contact, Trudgill proposes that whether inventories grow or shrink depends on the type of contact. He cites Nichols (1992) who argues that in situations of lengthy, stable contact between two languages, there are many opportunities for words containing non-native sounds to be borrowed, leading to growth in inventory size. For example, some Bantu languages acquired clicks through borrowing from neighbouring Khoe-San (Bostoen and Sands 2012).

A similar argument was offered by Haudricourt (1961), who suggested that bilingualism plays a role in inventory size. In particular in situations of “egalitarian” bilingualism, there
is a chance for inventories to grow larger as speakers borrow from each other’s languages.

On the other hand, Trudgill pointed to the simplifications that tend to occur in the
process of pidginization/creolization. A reduction in inventory size could be viewed as a kind
of simplification, and thus a possible outcome of language contact. In other words, contact
between languages could lead to growth due to borrowing, or loss due to pidginization.

The notion that pidginized/creolized languages should have smaller inventories is what
Klein (2006) calls the “creole simplicity hypothesis”. He tests this against a corpus of 23
creoles, and finds that it actually does not hold true of phonological inventory size. The
smallest inventory in this sample was Ndyuka (Hutter and Hutter 1994), with an inventory
of 19 sounds. Angolar (Lorenzino 1998) has the largest inventory, with 37 sounds. In other
words, Klein’s corpus of inventories has about the same range of sounds as most of the
languages in UPSID. On the other hand, Klein reports that creoles tend to use a narrower
range of stop contrasts than non-creoles.

Trudgill also explored the opposite case, that is, the development of languages in isola-
tion. His main focus was the Polynesian languages, which exhibit an interesting distribution
of inventory sizes: the size of the inventories shrinks as one goes from west to east across the
expanse where these languages are spoken. Inventories are somewhat larger toward Asia
and get smaller toward Hawaii. Trudgill poses the following question: “is there any connec-
tion between this geographical penetration deeper and deeper into the formerly uninhabited

The emphasis here is on “uninhabited”. The languages of the islands of the Pacific de-
veloped in relative isolation. They were geographically remote, and had contact mainly with
speakers of closely related Polynesian languages. Trudgill proposes the following possible
link between isolation and small consonant inventories: isolated communities are started
by a very small number of people, which leads to tighter social networks, which means that
speakers can assume more shared common ground between each other, which would eventually
lead to “a situation in which communication with a relatively low level of phonological
redundancy would have been relatively tolerable.” (Trudgill 2004, p. 315).

Community structure is the key part of Trudgill’s argument that makes it different
from Hay and Bauer (2007), or Atkinson (2011). It is not just about the number of people
speaking a language, it is about the network of speakers in a community. The way the
community network is organized depends to some degree on the size of the community, and
this is why there could be effects of population size on inventory size. In a later paper,
Trudgill (2011) makes it very clear he believes that “five social factors could be expected,
in combination, to have various kinds of influence on phoneme inventory size; it will never,
I suggest, be sufficient to look at population figures alone.”

Some of these arguments may be plausible, and Trudgill is probably correct that com-

munity structure has a non-trivial effect on language. However, the size of an inventory
is strongly influenced by sound change, something left unmentioned in these accounts. In
order for population size and inventory size to be tied to each other, sound changes would
have to “keep up with” the change in population size, e.g. more splits as the population

97
grows and more mergers/deletions as the population falls.

Moreover, it is not obvious how population size could affect any of the actual mechanisms that can lead to sound change. Consider a specific change: final devoicing of obstruents. This is due to articulatory factors that make it difficult to maintain voicing in final position, which leads to the production of obstruents which learners perceive to be voiceless, resulting in a sound change (e.g. Blevins (2006b)). In what way could the population size affect the probability of this occurring?

Drawing on the previous discussion, one might attempt to argue as follows: Final devoicing is probabilistic and does not occur in every utterance. The size of a population affects how many people a learner interacts with, which affects how many voiced tokens or voiceless tokens that learner experiences. With a larger population size, there should be a greater probability of a sound change occurring.

The problem with such an argument is that the probability of a learner hearing a devoiced obstruent does not crucially depend on how many people the learner interacts with. It depends on how many words in the lexicon contain a voiced obstruent in word-final position and the relative frequency of such words. These are features of the language that have no connection to population size. Granted, some words are more frequently used in certain communities, so learners in different communities may experience different frequency distributions of sounds in their input. However, words are arbitrary sound strings, so neither the size of population, nor the structure of the community, could have any influence of the frequency of voiced obstruents or their distribution in the lexicon. It is essentially up to chance which words happen to be most frequent in a given language and a given community, at a particular point in time.

Even if population size did directly influence the proportion of devoiced tokens that a learner experiences, this still would not provide a convincing link between population and inventory size, because the devoicing is not guaranteed to either increase or decrease the size of an inventory. Suppose an inventory only has /b/. Final devoicing of /b/ to [p] could lead to the introduction of a new phoneme /p/ into the inventory, increasing the size. If the inventory already had both /b/ and /p/, then the distinction becomes neutralized in final position, but there would be no change to overall inventory size. If for some reason /b/ only occurs in final position, then /b/ could merge with /p/, and inventory size drops. The same holds true for any other arbitrary sound change. Assuming the sound change is largely, or only, affected by misperception, I can see no reasonable connection between population size, community structure, and the probability that a sound change increases or decreases the cardinality of the inventory.

4.1.3 Hypothesis #1: Phonotactics and inventory size

There is another rather different correlation that I think can help tie together sound change and inventory size. Maddieson (2007) divided the languages of UPSID into three categories based on their phonotactics: simple (V or CV syllable shapes only), moderate (CV, VC,
CVC, and also CCVC but only if C2 is a glide) or complex (anything else). Comparing
the groups, Maddieson found a correlation between phonotactic complexity and inventory
size: languages with simpler phonotactics tend to have smaller inventories. Languages with
simple phonotactics had an average of 17.66 consonants, those with moderate phonotactics
had an average of 21.3 consonants, and languages in the complex group had an average of
25.8 consonants in their inventories.

This is easily seen at the extreme ends of inventory size: many of the smallest invento-
ries belong to Polynesian languages, which have very simple phonotactics, while languages
of the Caucasus, which have some of the largest inventories, also have much more com-
plex phonotactics. (The Khoe-San languages stand out as an anomaly here. They have
extremely large consonant inventories, but tend to have simple syllable structure.)

From the perspective of sound change through transmission error, this correlation makes
sense. As discussed in the first chapter, sound changes occur when learners misperceive or
misanalyze some part of the signal as something other than what the speaker intended (e.g.
Ogha (1981), Blevins (2004)). For instance, a voiced stop produced in final position may
lack some cues for voicing, leading a listener to misinterpret it as voiceless. Misperceptions
such as this are typically context-sensitive. Simpler phonotactics means a more highly
restricted set of contexts, which in turn means a smaller number of sound changes are
likely, or even possible. More complex phonotactics means more sounds come into contact
with each other, and there are more opportunities for more, and more different, kinds of
changes.

For example when a language has CV as its maximum syllable, consonants can only
appear in two kinds of environments: utterance initial, or intervocalic. Compare this to a
language that permits up to CCVCC, so that consonants can appear initially, finally, and
with either vowels or consonants on either side. There will be contextual effects on the
articulation and perception of consonants in this language that will never apply in the CV
language. To take the example of final devoicing again, this is a process that simply cannot
happen in a language that has no codas. This means that voiceless consonants cannot be
“created” in a CV language through final devoicing, while this can occur in a CVC language.

The more contexts, therefore, the more diverse possibilities for sound change, and in
the long run languages with more permissive phonotactics should develop larger inventories
than those with more restricted phonotactics. This is stated below as the first hypothesis
of this chapter, which will be tested by computer simulation in Chapter 5.

**Hypothesis #1** Inventory size is tied to phonotactic complexity, since sound change is
partly context-sensitive, and phonotactics defines the set of possible contexts in a language.
Languages with more permissive phonotactics should tend to eventually develop larger
inventories than those with more restrictive phonotactics.
4.2 Inventory contents

4.2.1 Overview

Inventories differ not only in how many segments they have, but of course also in terms of which segments. Sound inventories are extremely diverse, and there is no “universal” consonant that appears in all languages. Nasals are the most common type of sound. In UPSID (Maddieson and Precoda 1989) /n/ occurs in more inventories than any other, and in P-base (Mielke 2008) the most frequently occurring consonant is /m/. These nasals each appear in over 90% of languages in their respective databases. It is also interesting to note that no vowel is universal either. The most common one is a high front unrounded vowel /i/, appearing in more than 90% of the languages in both UPSID and P-base. The image in Figure 4.10 shows the pulmonic consonant portions of the International Phonetic Alphabet chart, with cells size warped by relative frequency of the consonants in P-base\(^1\). Note that consonants are not very evenly distributed. There are a few categories which have very large cells, but there are many more which are almost too small to see.

Figure 4.10: IPA chart warped to show consonant frequency in P-base (Mielke 2008)

Finding absolute universal properties of consonant inventories has proven difficult. Hyman (2008) proposed only four:

1. Every phonological system has stops

2. Every phonological system contrasts stops with non-stops

---

\(^1\)Credit goes to Jeff Mielke, the creator of P-base. This image can be generated using a visualization tool available on his website at [http://pbase.phon.chass.ncsu.edu/](http://pbase.phon.chass.ncsu.edu/)
3. Every phonological system uses place of articulation contrastively.

4. Every phonological system has coronal phonemes.

Shortly after Hyman’s article appeared, Blevins (2009) published a reply article suggesting that #4 may not actually be a universal. She argues that Northwest Mekeo, an Oceanic language, has no coronal phonemes. Blevins does report that surface coronals appear, however. The velar stops become palatalized coronal affricates before [i], and the velar nasal has a surface variant [n]. Coronals therefore are articulated by speakers of Mekeo, but the coronal place of articulation is not used contrastively in the language. However, Blevins also notes that there are borrowings that include the lateral [ɿ], and she does not argue that this [ɿ] is allophonic of anything, so perhaps coronal phonemes do exist in this language, even if only marginally.

Given the range of inventory sizes, as discussed in section 4.1, and the lack of universal consonants, it is unsurprising that there exist virtually no cases of languages with identical phoneme inventories (and it is probably impossible to find any with the same phonetic inventory). P-base contains a few identical inventories, but in some cases these belong to dialects of the same language: Kirzan Armenian and Standard Eastern Armenian are listed as having the same inventory, as are several varieties of Irish English. There were only four pairs of languages in P-base with identical inventories that were not listed as dialects, and three of them are Australian languages: Arabana and Wangkangurru, Garawa and Ganggulida, Wambaya and Nyulnyul. The only inventories found outside of Australia that were identical belong to two Indo-Aryan languages, Bagri and Marwari (Shekhawati dialect).

More matches can be found by narrowing the search so as to look for only identical consonant inventories, and excluding vowels. Bagri and Marwari are still the only Indo-European languages that turn up. Hiligaynon and Balangao, Austronesian languages spoken in the Philippines, also have identical consonant inventories. There was only a single pair of Afro-Asiatic languages with the same consonants: Hurza and Muyang, both Chadic languages of Cameroon. There was only a single pair of languages in the Americas with matching consonants: the Uto-Aztecan languages Comanche and Shoshoni. Finally, there were four sets of Australian languages with identical inventories: (1) Ganggulida, Garawa, (2) Biri, Ngiyamba, (3) Gunin, Wambaya, Nyulnyul, and (4) Arabana, Martuthunira, Muruwari, Wangkangurru, Yinggarda. It is worth noting that in every case, the languages are spoken in areas nearby one another, and they share linguistic ancestors. There are no completely unrelated languages in P-Base with identical inventories.

While it may be rare to find entirely identical inventories, there is another, similar, relationship that is more common: some inventories are supersets of others. In other words, there are many pairs of languages A and B, where A has all of the phonemes that B has, and then some. The scatter plot in Figure 4.11 shows the number of superset inventories for each inventory in P-base.
Figure 4.11: Consonant inventory size and number of superset inventories in P-base

There is a very clear trend: small inventories have many supersets, large inventories have none. From a numerical perspective, this makes sense. The bigger an inventory gets, the smaller the chances that there will be another, yet larger, inventory that can act as a superset. There is a drop off when inventories reach 20 consonants. After this, inventories rarely have any supersets at all. This suggests that as inventories become larger, there is an increasing amount of diversification that takes place. Put otherwise, large inventories tend to have all the sounds also found in small inventories, and then some more rare ones.

I searched P-base for the frequency of each consonant symbol, and found that there are 299 unique consonants in the database, that is, consonants that appear in only a single language (and some languages have more than one unique consonant). Figure 4.12 shows a plot of inventory size against the number of unique consonants, and the expected trend appears: larger inventories tend to have more unique consonants. Virtually all inventories above 40 have at least one unique sound.

The unique consonants are ones that would be regarded as “complex”, such as /ⁿdβ/ or /χ^yw/. On the other hand, P-base has 22 highly frequent consonants that are found in 200 or more languages each: /p, b, t, d, k, g, ?, f, v, s, z, j, tf, h, m, n, n̥, 1, r, j, w/, all of which would be considered “simple”.

It is important to point out that the superset relations are at least in part due to the fact that P-base contains related languages. If languages share a common ancestor, then it becomes more likely that they will have phonemes in common. For example, the inventory of Comanche is a subset of the inventory of Western Shoshoni, and both languages are listed in the Central Nomic group in P-base. Languages may also have phonemes in common if they are geographically close to each other. However, there are many examples of languages that are in a subset/superset relation, and which are neither genetically nor geographically close to one another. The inventory of Blackfoot (Algie, western Canada) is a subset of Tiv (Niger-Congo, Cameroon), Ainu (isolate, Japan), and Javanese (Austronesian,
Indonesia) among others. Ideally, comparisons of inventories would be done using a corpus of languages, with a balanced number of language families and locations, but this is a difficult task to accomplish, given the limited phonological data that is available.

A similar superset relationship has been found in the languages of UPSID, in a study by Lindblom and Maddieson (1988) (see also Maddieson (2011)). They grouped all of the consonants in UPSID into three sets, which were supposed to represent increasing consonant complexity. Set 1 they refer to as containing “basic articulations”, Set 2 contains “elaborated articulations” and Set 3 contains “combinations of elaborated articulations”. They begin by defining Set 2 consonants, then Set 3, leaving anything unspecified in Set 1.

Set 2 contains voiced obstruents and voiceless sonorants, which Lindblom and Maddieson say are “departures from the default mode of phonation”. Pre-nasalization, nasal release, and pre- and post-aspiration are also included in Set 2. Also included are ejectives, clicks, and implosives. Five places of articulation also belong in Set 2: labiodental, palato-alveolar, retroflex, uvular, and pharyngeal. Finally, palatalization, labialization, and pharyngealization are included in Set 2.

A sound can only belong to a single one of these categories to remain in Set 2. If it combines two or more, then it goes into Set 3. For instance, /d/ would be in Set 2, as a voiced obstruent, but /ð/ would be in Set 3 because it additionally has palatalization. The consonant /k/ belongs in Set 2, but /kw/ is in Set 3 because it is an ejective and it has labialization. Set 1 consists of the “leftovers”, any sounds not grouped into Set 2 or Set 3. Lindblom and Maddieson specifically list the following as Set 1 consonants in UPSID: /p, t, k, ?, s, h, m, n, ɳ, l, r, w, j/.

Lindblom and Maddieson found that all languages use at least some of the consonants in Set 1. Interestingly, they also found that there is a correlation between inventory size

Figure 4.12: Consonant inventory size in P-base and number of unique consonants
and the probability of having consonants from one of the three sets. Small inventories tend to be made up primarily of Set 1 consonants, and as inventories get bigger more Set 2 and Set 3 consonants are found.

I attempted to replicate Lindblom and Maddieson’s study using the inventories of P-base. It was somewhat challenging to decide which sounds from P-base should go into which Set. Lindblom and Maddieson described sounds in terms of IPA-style articular features. P-base defines all of its segments based on phonological features, and these do not all have a nice fit to articular descriptions. In the end, categorization was achieved through of a combined search of both phonological features and the descriptive names of the Unicode strings. Voiceless nasals, for example, are easily found with the feature set [+nasal, -voc, +son, -voice]. On the other hand, it turned out to be easier to find consonants with pharyngealization by looking for the existence of the name “MODIFIER LETTER SMALL REVERSED GLOTTAL STOP” in a Unicode character.

A simple Python script was created with a list of phonological feature values and Unicode names that should be flagged as belonging to the elaborated set (Set 2), based on the descriptions of Lindblom and Maddieson. For each segment in P-base, a score was assigned for the number of flagged descriptions. If a segment only matched one flag, then it was assigned to Set 2. If it matched more than one flag, it was assigned to Set 3. If it matched none of the flags, it was assigned to Set 1.

This actually yielded a very different collection for Set 1 than what was reported by Lindblom and Maddieson. For instance, doubly-articulated voiceless stops are not mentioned at all by Lindblom and Maddieson, so they by default end up in Set 1 since they match no other descriptions. Only a single voiceless affricate was in Set 1 in UPSID, but there are more than a dozen in P-base. Interdental fricatives are also not mentioned at by Lindblom and Maddieson. I flagged only the voiced ones, which matches the spirit of their proposal that voiced fricatives be considered Set 2 while voiceless fricatives belong in Set 1. Geminate consonants were not mentioned by Lindblom and Maddieson either, but I decided to flag them as well.

Despite some of these differences, the results for P-base are very similar to what Lindblom and Maddieson found for UPSID. Figure 4.13 shows a plot of inventory size against number of segments in each complexity class. For inventories larger than about 40 segments, the results are unsurprising. The cardinality of Set 1 is less than that of Set 2 or Set 3. As inventory size grows, it is only natural that the number of Set 1 consonants will be less than the number of Set 2 or 3 consonants. The interesting part of this figure is the lower left quadrant, with the smaller inventories. It is entirely plausible for these small inventories to consist mostly of Set 3 consonants, since this set has the largest number of members to draw from, but instead we find that Set 1 consonants are heavily represented.

There are two parts to Lindblom and Maddieson’s correlation: the frequency of sounds, and their articulatory complexity. I only want to focus on the former. In any case, it is questionable whether a coarse grouping into three sets is the best way to classify sounds by complexity. As I found in trying to replicate the analysis, there are many sounds which
Lindblom and Maddieson did not consider, and depending on how these other sounds get sorted the correlation might come out differently. Therefore, I do not think we can conclude much about the relative frequency of simple and complex sounds.

However, ignoring complexity, we are still left with the kind of superset relation that I found in P-base: there exists a small set of sounds which are common to inventories of all sizes, and inventories become increasingly diverse as they get larger.

Lindblom and Maddieson hypothesize that the superset relations are due to the way that inventory growth happens: inventories initially “saturate” the smaller space of easy-to-articulate sounds, and then move into more complex territory. This would explain why all inventories seem to make use of simple sounds from Set 1, and why only larger inventories tend to have Set 2 and Set 3 sounds. The authors use a metaphor of a rubber band and a magnet. Imagine a space of all sounds that could possibly be articulated. Somewhere in there is a small subspace of “neutral” sounds (i.e. those in Set 1). As consonant inventories grow, they first use up this space, then begin to expand beyond it into more complex sounds (Set 2 and Set 3). A metaphorical rubber band pulls the sounds back towards the simpler articulations, while a metaphorical magnet pushes sounds apart from each other.

They note that one way to refute this would be to look for inventories that reverse this pattern, that is, small inventories with mainly Set 3 consonants, or large inventories with mainly Set 1 consonants. I conducted a search of P-base for counter-examples, which turned up only a handful of languages.
Figure 4.14: Consonant inventories from P-base with “reversal” segment complexity

The smallest inventory with mainly Set 2 and Set 3 consonants belongs to Sinaugoro (Austronesian, Papua New Guinea). This language has 10 consonants from Sets 2/3 and only 6 from Set 1, and the complex sounds are all voiced obstruents (some of them are also labialized). Other small inventories included Central Ojibwe (Algonquian, Canada), Dagur (Altaic, China), and Pileni (Austronesian, Solomon Islands). Ojibwe has 10 consonants from Sets 2/3 and only 5 from Set 1, and the complex sounds are all geminates. Dagur has 11 from Sets 2/3 and only 5 from Set 1, with all the complex sounds coming from voiced obstruents, or aspirated voiceless stops. Finally, Pileni has 11 from Sets 2/3 and 5 from Set 1, and its complex sounds include aspirated stops, aspirated nasals, and voiced obstruents. The full consonant inventories of these languages are shown in Figure 4.14.

The other reverse pattern would be inventories that are large and have mainly Set 1 sounds. This is difficult to find, if only because of the limited number of Set 1 consonants. The closest inventory is that of Shilluk (Nilo-Sarahan, Sudan). Shilluk has quite a large inventory of 31 consonants, 11 from Set 1 and 10 each from Set 2 and 3, so it has more Set 1 consonants than either Set 2 or Set 3, but not combined. It has some aspirated stops, voiced stops, and geminate sonorants. No elaborated places of articulation are used and there are no implosives or ejectives.

Another, similar, statistical analysis of UPSID was undertaken by Choudhury et al. (2006). They constructed a bi-partite graph where nodes represented either languages or consonants. A language node was linked by an edge to a consonant node if that consonant was found in that language’s inventory. Choudhury et al. then looked at the so-called “degree distribution”. The degree of a node is simply the number of edges connected to it.

They found that the frequency of consonants follows a power law, where a small number of consonants are extremely frequent. They hypothesized a “principle of attachment”, where
consonant inventories grow by first selecting from the set of most common consonants before selecting from the less common ones.

4.2.2 Hypothesis #2 - Common consonants

If taken literally, Lindblom and Maddieson’s claim about inventory growth is questionable. They provide no historical evidence that languages actually do grow by expanding from a basic set, and no specific examples of sound change mechanisms were offered to substantiate the “rubber-band and magnet” metaphor. Choudhury et al.’s “principle of attachment” fairs no better, since it relies on the same unfounded assumption that languages first grow from a specific set of sounds, before diversifying.

I propose that the force drawing languages toward a common set of consonants comes from context-free sound changes, rather than context-sensitive ones. Smaller inventories tend to have simpler phonotactics (by Hypothesis #1), which means fewer overall context-sensitive changes can take place, limiting the potential sounds that could join the inventory, and giving context-free change a greater weight. Larger inventories tend to have more complex phonotactics, which means that they could potentially be affected by a wider variety of context-sensitive changes, in addition to the context-free ones. This leads to a situation where both small and large inventories have sounds in common (namely the ones that are the result of context-free changes), while larger inventories can also develop rare or unique sounds from context-sensitive changes.

Hypothesis #2

Common sounds exist because of context-free biases in transmission that affect all languages, regardless of phonotactics. Rarer segments are rare because they require more specific phonetic environments to appear, and these are more likely to exist in larger inventories, because larger inventories have more, and more different, phonetic contexts (by Hypothesis #1).

4.3 Inventory organization

4.3.1 Overview

Even though consonant inventories differ quite considerably from language to language, they are not random collections of sounds. As even casual observation of sound inventories will show, they tend to “line up” along different feature dimensions, rather than being spread randomly about. For instance, consider the inventories of Noon (Niger-Congo, Soukka 2000) and Tamazight (Afro-Asiatic, Abdel-Massih 1971), shown in Figure 4.15.

Noon’s stop system is nearly a perfect square. There are stops at four places of articulation: labial, alveolar, palatal, and velar (plus the lone glottal stop). At each place, stops can
be voiceless, voiced, pre-nasalized, nasal, or implosive. Only the velar implosive is missing from the language. This language is very stop-heavy, having only a few fricatives. The inventory of Ait Ayache Tamazight makes significant re-use of a small number of features. Nearly all consonants can be either short or long. Velar and uvular stops have labialized versions. Pharyngealization is contrastive on coronal consonants, as is length. There are still few gaps in this system: there is no long \(/z/\), the pharyngeal fricative has no length contrast, and the uvular fricatives have no labialized version.

![Consonant inventories of Noon and Tamazight](image)

Figure 4.15: Consonant inventories of Noon and Tamazight

In comparison, consider the randomly generated inventory in 4.16. In this, sounds are spread more widely around the chart, and there is less re-use of a given feature. Certainly there is some re-use that occurs: there happen to be quite a few labial sounds, and the velars are well populated too. There are also cases of poor feature re-use, like the three voiced alveolar stops, which are the only instances of labialization, palatalization, or implosives in the entire inventory. There are also very few pairs of sounds that differ by a single feature, which is common in natural languages. Most pairs of sounds differ by more than one. This inventory also looks unnatural because it is small, yet has numerous complex or rare sounds, and lacks many of the simple common sounds.

![Randomly generated consonant inventory](image)

Figure 4.16: Randomly generated consonant inventory
This observation about the re-use of features was formalized into the concept of feature economy by Clements (2003). Clements was not the first to discuss the idea, however. Ohala (1992) suggested that consonant inventories tend to obey a principle of “maximal utilization of the available distinctive features”. Martinet (1952) referred to this as “the theory of pattern attraction” and, assuming functionalist principles, reasoned that having fewer articulations (i.e. distinctive features) made sounds more distinct, which made language easier to perceive and possible to produce. Clements (2003, p. 292) himself cites even earlier work by de Groot (1931, p. 121), discussing sound change: “those phonemes that appear are those which have only phoneme marks already figuring in the system”. The concept also relates to the findings of Lindblom and Maddieson (1988), discussed in the previous section. They proposed that languages first make considerable re-use of a small set of articulations before growing, while a metaphorical “rubber band” effect prevents inventories from splitting too far apart. In more recent literature, two additional measurements of economy were proposed by Hall (2007) and another by Mackie and Mielke (2011). Mackie and Mielke further showed that inventories of natural languages differ from randomly generated sets of segments, in terms of feature economy scores.

4.3.2 Feature economy

Clements (2003) defines the term feature economy as “the tendency for languages to maximize the ratio of segments over features”. He is careful to contrast this notion of economy with others. Feature economy in this sense is not the same thing as parsimony or simplicity, which would suggest that, all other things being equal, a smaller inventory, one without too many “parts”, is better than a larger inventory. Feature economy says that a language will maximize the segments it has, given a particular set of contrasting features. The absolute value of the size of the inventory is not what is important.

This is an important distinction to make, since the actual size of the inventories of the world’s languages varies enormously, as discussed in a previous section. Feature economy does not presuppose anything about inventory size, and a range of sizes is compatible with its predictions, whereas a simplicity-driven account of inventory structure is clearly at odds with the data, which does not suggest any tendency for language to economize on the sheer number of segments. It does, however, turn out that different measurements of economy are biased towards different sizes of inventories; this is discussed more in section 4.3.2.1.

Feature economy is a description of how sounds in a language relate to each other, and it should also be distinguished from economy in the representation of features, e.g. some kind of underspecification, which involves theoretical assumptions about the mental representation of features and lexical items (Steriade 1995, Lahiri and Reetz 2010).

Finally, feature economy should be contrasted with symmetry, although economical inventories do display a certain degree of symmetry. Clements illustrates this point with three examples of inventories, shown in Figure 4.17.
Assume that these example systems can be analyzed with only two features, [voice] and [continuant]. Place features are also required, but they are ignored here because all three systems make the same place contrasts. System A is completely symmetrical, and would be considered perfectly economical. Recall that Clements defines feature economy as maximization of the segment to feature ratio. In System A, this ratio is at its maximum and every combination of [voice] and [continuant] is in use at every place of articulation. System B is also symmetrical, but is not as economical, since there are no [+continuant, +voice] segments. System C lacks symmetry. There is a gap in the [+continuant, –voice] series, and two gaps in the [+continuant, +voice] series. Nonetheless, System C has greater economy than System B, because System C contrasts 13 segments to System B’s 12, both of them using the same number of features. Thus System C’s segment/features ratio is higher.

Clements (2003) present two specific predictions of the feature economy hypothesis. The first prediction is what he calls mutual attraction: “a given speech sound will occur more frequently in systems in which all of its features are distinctively present in other sounds” (p.296). The second is called avoidance of isolated sounds: “a given speech sound will have lower than expected frequency in systems in which one or more of its features are not distinctively present in other sounds” (p.306).

To turn to specifics, consider a voiced labial sound, call it V, which would be represented as [+voice], [+continuant], and [labial] in Clements’ feature system. According to the predictions of mutual attraction, V should be more common in inventories that also have another labial, another voiced sound, and another continuant. Clements clearly specifies that all three conditions would have to be satisfied for mutual attraction to be supported. It does not necessarily have to be the case, however, that there be three other sounds. Mutual attraction would be supported by inventories with V, P (a voiceless labial stop) and Z (a voiced coronal fricative).

To check this prediction, Clements analyzed the languages of UPSID. Here I summarize his findings for the co-occurrence of V and Z. The method he used is to compare the observed vs. expected co-occurrence frequencies of different segment pairs within inventories. In this case, that involved counting how many language have a V but not a Z, how many have a Z but not a V, how many have a Z and V, and how many have neither segment. The table in Table 4.1 comes from Clements (2003, p. 303) and shows the results for V and Z in UPSID.
Table 4.1: Co-occurrence of V and Z in UPSID (from Clements (2003, p. 303) )

<table>
<thead>
<tr>
<th></th>
<th>present</th>
<th>absent</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>present</td>
<td>110 (57)</td>
<td>37 (90)</td>
<td>147</td>
</tr>
<tr>
<td>absent</td>
<td>65 (118)</td>
<td>239 (186)</td>
<td>304</td>
</tr>
<tr>
<td>total</td>
<td>175</td>
<td>276</td>
<td>451</td>
</tr>
</tbody>
</table>

The numbers outside the parentheses in Table 4.1 are the actual occurrences of the segments. The numbers in parentheses are the expected frequencies, which are calculated on the assumption that the frequency of V or Z in a given cell of the table is proportional to its frequency in the sample as a whole. The expected frequencies are calculated as the sum of the row multiplied by the sum of the column that the cell is in, divided by the total number of languages.

If the prediction of mutual attraction is correct, then the actual frequency of languages with both V and Z is going to be higher than the expected frequency, i.e. the ratio of observed/expected frequencies is going to be greater than 1. Similarly, if the prediction of avoidance of isolated sounds is correct, then the cells corresponding to languages with only V or only Z are going to have a ratio of observed/expected of less than 1. As Figure 4.1 shows, the actual differences between observed and expected frequencies are in the direction of these predictions. Clements (2003, p. 304) reports that the differences are significantly different, and highly so ($\chi^2 = 119.203, p < 0.0001$).

To illustrate the second prediction, avoidance of isolated sounds, consider an inventory with a voiceless labial stop P and a voiceless coronal stop T. Such an inventory can be contrasted with the feature [labial] alone. Adding in a voiced labial stop B to this inventory requires adding a new feature, [voice], to contrast P and B. However, this is not as economical as having both B and a voiced coronal stop D. Avoidance of isolated sounds predicts that the observed/expected ratio for languages with B and D will be higher than expected and B without D will be lower than expected.

In addition to these reported correlations, Clements’ paper also proposes a way of calculating a feature economy score for an inventory. However, he calculated the economy of only three languages. Later work by Hall (2007) proposed two other metrics, but again only using the same three languages as Clements. Mackie and Mielke (2011) used these three metrics, plus one more, to calculate the economy of nearly 500 languages in P-base. These metrics are described in the next section.
4.3.2.1 Measuring feature economy

Clements (2003) offers a particular measurement of feature economy, which I will refer to as the Simple Ratio measurement. Using this measurement, the economy value $E$ is calculated as:

$$E_{S.R.} = \frac{S}{F}$$

where $S$ is the number of segments in the inventory, and $F$ is the minimum number of features required to contrast them all. The term “contrast” in this case has a particular meaning. Normally, saying that two segments contrast with each other suggests that there is a minimal or near-minimal pair in the language based on these segments. For the purposes of evaluating feature economy, segments are said to be contrastive if they differ by at least one feature.

Clements did not offer any specific methods for finding this minimum set of features, although some algorithms have since been proposed. The Feature Economist algorithm in Mackie and Mielke (2011) works by doing a pairwise comparison of every segment in an inventory checking for contrast. On the first pass, it checks using fully specified segments, then it discards increasingly larger sets of features until contrast becomes impossible. It is discussed in more detail later in this section.

The Successive Division Algorithm (Dresher 2003) is another way of finding contrastive features. This algorithm starts by selecting a feature at random, then it tries to divide the inventory into groups of segments that are either $[+F]$ or $[-F]$ (assuming binary features). Within each of these groups, a new feature is selected, and the segments again divided into $[+F]$ and $[-F]$ groups. This continues until the inventory has been divided up into groups consisting each of a single segment, and the contrastive features are whatever features were necessary to create these divisions.

Obviously, finding $F$ depends on which feature system is in use, so the score is always relative to some feature set. (In fact, Clements suggests that feature economy could even be used as a way of comparing different feature theories.) In Clements (2003), the feature system is a custom set of features that he selects for the paper, without any justification provided. They are [sonorant, labial, dorsal, nasal, voice, spread glottis, constricted glottis, continuant, posterior, apical, lateral]. To demonstrate how to calculate economy, Clements uses the inventories of Hawaiian, French and Nepali (see Figure 4.18).
a. Hawaiian: 8 consonants
(after Elbert & Pokai 1979)
\[ \begin{array}{cccc}
p & t & k \\
m & n & l \\
w & t & s \\
h & & & \\
\end{array} \]

b. French: 18 consonants
(after Dell 1985)
\[ \begin{array}{cccc}
p & t & \kappa \\
m & n & \lambda \\
\varphi & \delta & \xi \\
l & & & \\
\end{array} \]

Nepali: 27 consonants
(after Bandhu et al. 1971)
\[ \begin{array}{cccc}
p & t & ts & k \\
pb & db & tsb & lb \\
b & d & dz & l \\
\beta & \delta & \delta & \gamma \\
m & n & (o) \\
l & r & & \\
\end{array} \]

Figure 4.18: Inventories of Hawaiian, French and Nepali (from Clements (2003, p. 288)). Dashed boxes represent areas of the inventory that Clements' considered representative of feature economy effects.

Hawaiian has 8 consonants and is analyzed as requiring 5 features for contrast: [sonorant, labial, nasal, spread glottis, constricted glottis]. Given this, the economy value is \( E = \frac{8}{5} = 1.6 \). Clements is not explicit about how he comes to this minimal set. Whatever the method used, the minimal features are those for which not all segments in the inventory have the same value. For instance, Hawaiian has no lateral consonants, so every consonant is [−lateral], and therefore [lateral] is a non-contrastive feature.

French has 18 consonants and needs 7 features: [sonorant, labial, dorsal, nasal, voice, continuant, posterior]. Its economy value is \( E = \frac{18}{7} = 2.57 \). Finally, Nepali has 27 consonants and needs 10 features to be contrasted, that is, every feature except for [constricted glottis]. It scores the highest economy value of \( E = \frac{27}{10} = 2.7 \).

The maximum number of segments an inventory can have is \( n^F \) for a system with \( F \) \( n \)-ary features. Since phonological feature systems are usually binary, the maximum is usually \( 2^F \). The maximum economy score is therefore going to be reached when \( S = 2^F \). Practically speaking, it is unlikely for any language to reach this maximum, even for small numbers of features. Having ten features means it is possible to contrast up to 1,024 segments, an impossible size for a natural language inventory.

Clements (2003, p. 289) intends the metric to measure economy so that “the higher the value of \( E \), the greater the economy”. This turns out not to be the case, however, and the simple ratio measurement is biased toward larger inventories, meaning that larger inventories tend to get assigned higher scores. For example, a 32 segment inventory that can be contrasted with only 5 features is perfectly economical, since \( 2^5 = 32 \). Its economy score is \( E = \frac{32}{5} = 6.4 \). On the other hand, a 64 segment inventory that can be contrasted with 6 features is also perfect, because \( 2^6 = 64 \). The larger inventory has an economy score of \( E = \frac{64}{6} = 10.666 \ldots \). The larger inventory appears to be more economical, but both
inventories are actually as economical as they can be for their size.

More importantly, large imperfect inventories can have higher scores than small perfect inventories, which should not be the case if this number is truly measuring economy. For example, a 16 segment inventory that takes 4 features for contrast is perfect, since $2^4 = 16$. Its economy score is $E = 16/4 = 4.0$. On the other hand, a 50 segment inventory requiring 7 features would have an economy score of $E = 50/7 = 7.142$. In fact, for 7 features, the maximum inventory size is $2^7 = 128$, so a 50 segment inventory is not nearly as economical as it could be.

This is just a consequence of using a simple ratio as a measurement: a linear increase in the number of features results in an exponential increase in the number of possible segments. This results in inventory size being tied to economy score, firstly in the trivial sense of perfect scores necessarily being a power of 2 (or whatever base the feature system uses), but more significantly in that larger inventories can achieve higher scores that are impossible for smaller inventories.

Hall (2007) proposed two alternative measures, which he termed Exploitation and Frugality. The Exploitation metric measures how close an inventory comes to having the maximum number of segments, given the feature set. It is calculated as:

$$E_{\text{exploitation}} = S/2^F$$  \hspace{1cm} (4.2)

The other measurement proposed by Hall is called Frugality, and it is in some sense the mirror image of Exploitation. The Frugality metric measures how close an inventory comes to having the minimum feature set, given the size of its inventory. It is calculated as:

$$E_{\text{frugality}} = \frac{\log_2 S}{F}$$  \hspace{1cm} (4.3)

Both of these metrics have an advantage over the Simple Ratio metric, which is that they are bounded between 0 and 1, where 1 is a perfectly economical inventory. This eliminates the problem with Simple Ratio where differently sized perfect inventories could have different scores. This means that large inventories cannot outscore smaller inventories simply by being large.

In fact, large inventories will tend to be punished slightly under Hall’s two metrics. This is because the minimum feature set for a large inventory is going to be larger than the minimum feature set for a small inventory. A language with 50 segments requires at least 6 features, because $2^6 > 50 > 2^5$, while an inventory of 15 segments can potentially be contrasted with as few as 4 features, because $2^4 > 15 > 2^3$.

Inventories of these sizes would be assigned different scores on these metrics, even though both have the minimum possible feature count given their size. The large 50-segment inventory has a maximum Frugality score of $\frac{\log_{250}}{6} = 0.940$, and the smaller 15-segment inventory has a maximum Frugality of $\frac{\log_{15}}{4} = 0.976$. For Exploitation, the 50-segment inventory scores $\frac{50}{2^5} = 0.781$ and the smaller 15-segment inventory scores $\frac{15}{2^4} = 0.9375$. 

114
Mackie and Mielke (2011) proposed yet a fourth metric, called Relative Efficiency, that attempts to address this problem. This metric looks at how close an inventory is to the minimum feature count, relative to what the minimum and maximum counts would be for an inventory of that size. It is calculated as:

$$E_{R.E.} = 1 - \frac{F - F_{\text{min}}}{F_{\text{max}} - F_{\text{min}}}$$

(4.4)

Where $F$ is the minimum number of features actually required to contrast the inventory, $F_{\text{min}}$ is the minimum possible number for an inventory of this size, and $F_{\text{max}}$ is the maximum number of features. More specifically, $F_{\text{min}} = \lceil \log_2 S \rceil$, meaning that $\log_2 S$ is rounded up to the next integer, and $F_{\text{max}} = S - 1$. When an inventory is maximally economical, then $F = F_{\text{min}}$, and the term inside the square root evaluates to 0, which gives an overall score of 1. Relative Efficiency will assign the same score to a 50 segment inventory with 6 features as it will to an inventory of 15 segments with 4 features because in both cases the numerator evaluates to 0 (unlike Frugality and Exploitation, as shown above). When an inventory is maximally uneconomical, then $F = F_{\text{max}}$ and the numerator and denominator are therefore equal, so the inside term evaluates to 1, which gives an economy score of 0.

### 4.3.3 Cross-linguistic tendencies

Mackie and Mielke (2011) measured the feature economy of 479 languages in P-base using all four of the metrics discussed above. The segments in P-base are all fully specified using a modified version of the Sound Pattern of English feature system (Chomsky and Halle (1968), see Mielke (2008)). To determine the minimum number of features required for a given inventory, the Feature Economist algorithm was used. An overview of this algorithm can be found in Mackie and Mielke (2011). In section 5.4.4 there is a more technical description of the implementation of Feature Economist used in this dissertation.

Figure 4.19 gives the distribution of scores calculated in the languages of P-base by Mackie and Mielke (2011) using this algorithm. Exploitation is shown on a logarithmic scale because the small range of scores is more clearly displayed this way. It should be noted that there is a general problem with the statistics reported by Mackie and Mielke, which is that the languages of P-base are not independent. Many of the languages are related to each other, and since related languages have similar phoneme inventories, they will also have similar feature economy values. Additionally, languages spoken in geographically close regions may borrow phonemes from each other, resulting in similar inventories and similar economy scores. Ideally, economy would be calculated over a set balanced by languages family and geographical location.
Of the languages in P-base, West Greenlandic (Fortescue 1984) scored the highest on all metrics except Simple Ratio. The inventory is shown in Table 4.2. This language has 17 segments, and requires 5 features. This gives it a very high Frugality score of 0.817, because 5 features is the minimum possible for a 17 segment inventory. It scores even better on Relative Efficiency, reaching the maximum score of 1.

The maximum number of segments for 5 features is 32, so the Exploitation score is only 0.531, but this is still the highest score achieved in the entire sample of languages. This is because the maximum of 32 is a relatively small number, within the range of the majority of inventories in P-base. As the number of features climbs, the maximum inventory size grows exponentially, making it increasingly harder to get a high Exploitation score. The maximum size for a 7 feature inventory is 128 segments, which is already larger than that of practically any language on Earth. It is likely that no language could have an inventory large enough to reach a perfect Exploitation score for 8 or more features. In this way, Exploitation is the reverse of Simple Ratio, because it tends to assign higher scores to smaller inventories. (This difference in scores was discussed in some detail at the end of the previous section.)
Table 4.2: The inventory of West Greenlandic

Indeed, it is obvious that languages will very rarely attain perfect scores on any of these economy metrics, with West Greenlandic being the outlier in this case. There are reasons other than pure economy that constrain the shape of inventories. To attain a large Frugality or Exploitation score, every single feature value combination must be used, and in some cases this is not possible, or at least unlikely. For instance, it is common for obstruents to contrast in voicing, but not sonorants. This means that in a system that requires both [sonorant] and [voice], there is probably not going to be maximum use of features, and the combination [+son, −voice] has forces acting against it that are unrelated to economy. Some feature combinations may be literally impossible, such as [+high, +low].

If feature economy is an organizational principle of language, whatever its basis, then we would expect that randomly generated sets of segments will not show the same economy effect. Mackie and Mielke (2011) tested this prediction by creating randomly generated sets of segments, and calculating their economy scores. For each inventory in P-base, a new inventory of equal size was generated. Segments were added to each inventory by drawing randomly from a pool of all the segments in P-base. The probability of selecting a particular segment was equal to its relative frequency in P-base.

Of the 479 inventories generated, 121 were “non-contrastive”, meaning that they were generated with segments containing identical feature specifications. This is a limitation that results from the fact that P-base uses modified SPE features which do not fully distinguish between all of its segments. For instance, /tʃ/ and /tʃ/ are given exactly the same specifications, namely [+cons, −voc, −son, −cont, −voice, −nasal, +cor, −ant, +strid, −lat, −back, −low, +high, −round, +distr, ncovered, −syl, +tense, +del_rel, ndel_rel_2, −glot_cl, +hi_subgl_pr, nmy_glot_cl, −LONG, −EXTRA]. If a language happens to have both of these segments, it is “non-contrastive”, because there is no way to contrast these two elements of the inventory. This means it is not possible to calculate feature economy of these inventories: the feature-reduction algorithm will fail before even starting, since it will not be able to find contrast on even the first step. Only 358 of the 479 inventories generated were “contrastive” and subjected to analysis. The results are given in
Figure 4.20: Feature economy scores of natural languages and randomly generated inventories

Mackie and Mielke performed an ANOVA with type (natural vs. random) and inventory size as factors for each of these metrics. Main effects were significant for all four metrics. The interaction was significant with all except Exploitation. Inventory size and economy are positively correlated when using the Simple Ratio or Relative Efficiency measures, meaning that larger inventories have higher economy scores. The correlation is negative for Frugality and Exploitation, meaning that larger inventories have lower economy scores. The reasons for these correlations were discussed in some detail in section 4.3.2.1. Natural language inventories tended to score higher on all four metrics compared to randomly generated inventories.

Coupé et al. (2011) have, in contrast, argued that feature economy is not a very strong tendency, based on their own analysis of UPSID. Their methodology differs considerably from the one used by Mackie and Mielke (2011). Rather than using a small set of phonological features, Coupé et al. used a set of 100 phonetic features based on the labels used in the standard IPA chart.

The authors use what seems to be an Exploitation-type measurement, measuring actual
segments compared to possible segments, although they do not cite the Exploitation metric defined by Hall (2007) (they also conflate economy with ease of articulation, but this might be forgiven considering that they are using only articulatory features).

“If feature economy (ease of articulation) were to be the only principle acting on the content of PI [phonological inventories], we would expect systems to show a maximal use of features. In other words, the ratio of the actual number of segments in a system by the number of possible segments (given the set of features of this system) should be close to 1.” (Coupé et al. 2011, section 3.2)

The authors compared economy scores in inventories when segments are specified for all features, to the economy of inventories if only minimal specifications are used. Unsurprisingly, if no feature reduction algorithm is used (i.e. every segment gets full specification), then inventories are very uneconomical. Given the 100 features being used, an inventory needs $2^{100}$ segments to reach maximum economy.

To find the minimum feature set, Coupé et al. used a method that is not clearly described, and is questionably efficient: “We developed an algorithm calculating all the possible minimal underspecifications of a system. It relies on a massive test of acceptable descriptions with subsets of features (tens of millions for the largest system)” (Coupé et al. 2011, p. 2). The Feature Economist algorithm described in Section 5.4.4 generates around 100,000 feature subsets over the course of analyzing a single inventory. Part of the reason that Coupé et al. generate so many more sets is because they use a massive 100 features, whereas the feature system used in PyILM has only 19 features.

It is not clear why the authors chose to look for every possible minimal underspecification. For the purposes of feature economy, it is not necessary to know how many underspecified feature sets can be used to contrast an inventory. It is only necessary to know how small the smallest of these sets is. It also does not matter if there are multiple smallest possible sets, because we only need the cardinality of the set to do a feature economy calculation. The specific members of the set are irrelevant.

In absence of a full description of the methods in Coupé et al. (2011), and given the more detailed results in Mackie and Mielke (2011), it can be concluded that natural languages inventories are more economical than would be expected by chance alone. That is, feature economy is a property of phonological inventories of human languages.

### 4.3.4 Explaining economy

Very little work has been done examining the diachronic origins of feature economy. Virtually all of the existing literature relates to the synchronic phenomenon, although very recently some work has appeared that attempts to address diachrony.
4.3.4.1 A computational model

Pater and Staubs (2013) appear to be the first to attack the issue, which they call an unsolvable problem for phonology. They present an agent-based computational model for the emergence of feature economy. The model uses only two agents, and a small artificial language. The language consists of 6 meanings, and each meaning is paired with a set of 3 possible phonological forms. Meaning 1 and Meaning 2 can be pronounced as any of [pi], [bi], [pʰi], Meaning 3 and Meaning 4 can be pronounced as any of [di],[ti],[tʰi], and Meaning 5 and Meaning 6 can be pronounced as any of [gi],[ki],[kʰi]. Learning agents use constraints that map meanings to phonological forms. For example, a constraint could be $M1 \rightarrow [pi]$, which should be read as "Meaning 1 is associated with the surface form [pi]." Constraints are ranked and violable, and agents produce language by selecting a meaning, then selecting the surface form demanded by the highest ranked constraint that references the selected meaning.

The constraint ranking is not given in advance, but learned by each agent. Learning agents are only given a surface form, and must infer the meaning of the word on their own. Agents use a technique called Robust Interpretive Parsing (Tesar and Smolensky (1998)) to do this. This learning technique can be simply summarized as follows: The listener updates their grammar, by promoting and/or demoting certain constraints, if the pronunciation it would have chosen for the inferred meaning does not match the one in the observed output.

For example, suppose the speaker produces [bi] with the meaning M1. The learner hears [bi], but does not know the meaning. There are only two possible meanings for [bi] in this simulation (M1 or M2), and the learner will guess one of them. Let us say the learner picks M2. She then checks what her production grammar would generate for M2, and if that word is [bi], the learner promotes the constraint $M2 \rightarrow [bi]$ (because it matches the input, even though she is wrong about the meaning). Then the learner demotes the constraint that has the same output, but the alternative meaning. In this example, it would be $M1 \rightarrow [bi]$ that gets demoted. If the grammar would have generated something different for M2, like [pʰi], then learner should demote $M2 \rightarrow [bi]$.

This has the effect that a simulated lexicon will come to have categorical pronunciations for words, i.e. meanings will be consistently paired with single pronunciations which differ from the pronunciations of other meanings. The alternative outcome would be lexicons where every meaning has a multiple possible pronunciations and one is chosen with a certain probability on each utterance.

In fact, categorical pronunciation was the result in 9979/10000 simulations. Given the parameters of the simulation, there are really only three ways of achieving categorical pronunciation, because there will always be two words with a labial (Meanings 1 and 2), two words with a coronal (Meanings 3 and 4), and two words with a velar (Meanings 5 and 6).

The first possibility is that the final lexicon can be contrasted using these place features and only one other feature e.g. the lexicon is [pʰ, bi, ti, di, ki, gi], where [voice] is the
only contrast used at each place. As far as feature economy is concerned, this is the most economical outcome because of the low number of features required.

The second possibility is that one feature will be used for contrast at one place of articulation, and a different feature will be used at the other two places. This is less economical than the first possibility, and an example lexicon would be something like [pi, pʰi, ti, tʰi, gi, ki]. This set requires the place features, in addition to the features [voice] and [aspirated], which are not used to their full potential (velars are the only set to use [voice]).

Finally, the third outcome, which Pater and Staubs consider to be the least economical, is one where each place would have a different feature contrast, e.g. [pi, pʰi, ti, di, gi, kʰi].

It is not clear why three contrasts are considered less economical than two, given the very limited parameters of the simulation. Be there 2 or 3 contrasts, the inventory still requires both the features [voice] and [aspirated]. As far as calculating economy goes, both kinds of inventories would score the same, because they both have 6 segments and 3 features.

In cases of 1 or 2 contrasts, segments at the same place of articulation differ by only one feature. What makes the case of 3 contrasts different is that there will be one pair of segments that differ by both [voice] and [aspirated], e.g. /kʰi/ vs. /gi/. The inventory is highly restricted, so the 3 contrast cases are unfairly penalized. There is no way for [+voice, +aspirated] sounds to occur in this simulation, so there’s no way for /kʰi/ to get a “matching” /gʰi/ and increase economy.

Results from Pater and Staubs’ simulations are shown in Table 4.3. The number of contrasts refers to the number of features required beyond the place features. An example of a “1 contrast” outcome would be one where the final learner’s lexicon is [pi, bi, ti, di, ki, gi], so that [voice] is the only necessary feature. This is also the most economical outcome. By chance, this outcome is only expected about one tenth of the time, but it occurred one ninth of the time in the simulations. Pater and Staubs interpret this as a preference for economy.

The authors argue that this shows how feature economy can emerge without the need for any constraints that specifically encourage it. Instead, feature economy is the result of
learners preferring categorical pronunciations for words. This tendency in turn is a result of using Robust Interpretive Parsing. This model is an interesting attempt to account for feature economy from a diachronic perspective, but it falls short in several respects.

First, the results are not very generalizable. They rely on some very strong assumptions, namely Robust Interpretive Parsing, and a highly specific constraint set. The constraints are also very unusual for phonology in that they map meanings to sounds directly. Typical phonological constraints refer only to phonological (and perhaps morphological) material. This leads to a kind of non-Saussurean model of language. Sounds and meanings have a necessary connection, rather than an arbitrary one; agents know that if the word has a labial in it, then it must either convey Meaning 1 or Meaning 2, and the same with coronal and velar sounds, mutatis mutandis.

Second, sound change is not realistically modeled. What happens is that the pronunciation of words shifts randomly over time, within a small defined space of possible pronunciations. Meaning 1 may start out as [pi] and turn out as [bi] on one run of the simulation, [pʰi] on another but stay [pi] on another. This is not a true reflection of how sound change happens. The probability of /p/ voicing to [b] in word-initial position before a high front vowel is not the same as the probability that it aspirates in this position, nor is there an equal chance of it remaining a [p]. The most likely outcome is aspiration, since aspiration tends to be longer before high vowels (Klatt 1975). There is also no possibility for changes to cross the place of articulation boundary due to the way that constraints work (e.g. Meaning 1 could never be paired with /ta/ or /ka/).

The role of sound change in feature economy is an important issue to address. How can languages tend toward economy, all the while undergoing (phonetically-motivated) sound changes?

4.3.4.2 Whistle experiments

A very different sort of experiment that attempted to shed light on feature economy was undertaken by Verhoef and de Boer (2011). In this experiment, participants were taught 12 different signals on a slide whistle. The slide whistle was chosen because it allows for the production of a large number of sounds, requires little training, and it is unlikely that there would be interference from the participants’ first language.

Participants underwent a very short training session, hearing each signal only 4 times. At the end of the training session, participants were required to play all 12 signals to the best of their abilities. If they could not recall them correctly, then they were to play whatever they could remember. Learning was entirely unsupervised, meaning that participants were never told if they were correctly or incorrectly reproducing a whistle.

The experiment begins with a single participant, who learns from some whistles recorded by the experimenters. The whistles that are produced as responses during the testing session are recorded and used as the target whistles for the second participant. The second participant is unaware that their learning data was produced by a previous participant.
The process is repeated with the second participant, and their responses are provided to the third participant, and so on.

Errors were common during the testing stage, but participants were not corrected, and these errors were "passed on" to the next participants. The final set of whistles in the experiment looks quite different from the initial set. This alone is unsurprising. The interesting outcome is that whistles produced at later points in the experiment seem to consist of "sub-parts" that are re-used across multiple whistles. Verhoef and de Boer were able in some cases to trace back the ancestry of some of these sub-parts, as shown in Figure 4.21.

The reason that the re-use of sub-parts occurs is because the task of remembering 12 arbitrary whistle signals, with only 4 exposures each, is difficult. Participants often forgot what a signal was, but were forced to produce a signal regardless, so they simply played anything at all that they could remember (i.e. they played parts of other whistles they actually did learn correctly). This has the effect of introducing signals with shared sub-parts into the input to the second generation, even though those similarities did not exist in the input to the first generation.

The second participant will also not learn all 12 signals, and will rely on the same strategy of playing anything they do remember. This introduces more subwhistles into the input, or possibly spreads the previous subwhistle to new signals. By the end of a chain of learners, it appears as though there is a set of sub-parts that can be recombined to make a signal, though that was not a feature of the initial set of signals in the chain.

Verhoef and de Boer speculate that these subwhistles are analogous to features in natural language, and that what happened in the experiment parallels the development of feature economy: "the formation of building blocks here does not resemble the simplest dispersal models, but is more reminiscent of the 'Maximal Utilisation of Available Distinctive Features' principle proposed by Ohala or 'feature economy'. If a building block is present,
it tends to get mirrored and reused before new ones appear” (Verhoeof and de Boer 2011, p. 2069).

The results depend on the participants at some point failing to learn the signals properly, and inventing something new (which contains old, re-used parts). One reason that participants had difficulty remembering whistles is that they are completely arbitrary sequences of sounds. This is quite unlike natural languages, where phonemes are meaningless, but sequences of phonemes (i.e. morphemes) are meaningful. Perhaps if the whistle signals were paired with meanings, then the task of learning them would be easier.

This was addressed in a later experiment in Verhoeof et al. (2013). In this study, the authors trained participants to learn signals on a slide whistle that were paired with images of objects. In one condition, the whistles produced by one participant were used to train the next participant, exactly as in the previous study. This was the “intact” condition. In the “scrambled” condition, the experimenters randomly re-associated whistles from one participant with different objects, and the next participant was trained on this pairing.

Verhoeof et al. (2013) found no significant differences between the groups. The final sets of whistles in both conditions showed the same kind of re-use of sub-parts. One interesting difference that the authors noticed was how soon this re-use started to occur. For this they created a method for segmenting the whistles and calculating entropy. Entropy is a measure of uncertainty, and in this case entropy is used to measure differences between whistles – high entropy values indicate whistles that are less similar to each other (see Verhoeof (2012) for details on how this calculation was done). They conclude: “the main ‘drop’ in entropy in the intact condition took place approximately from generation four to eight, while in the scrambled condition, this was sooner, approximately from generation one to five” (Verhoeof et al. 2013, p. 3674).

This experiment is quite interesting, and it is in line with previous results in the iterated learning literature. Early results came from Kirby (2000) who produced agent-based computational models for the emergence of compositional syntax from non-compositional language (see discussion in Chapter 1, section 1.2).

Still, the whistle experiments do not directly touch on speech production or perception, so it is difficult to say if the results apply to language or not. The crucial claim is that the sub-parts of a whistle are comparable to features in language. However, it seems like Verhoeof and de Boer (2011) are intending each whistle signal to be analogous to an entire word. If this is the case, then re-use of sub-parts makes those sub-parts more like consonants or vowels, not features, so talk of economy or maximal re-use of features is not entirely relevant.

Even treating the entire whistle signal as one very long sound, there is the problem that these re-used sub-parts are sequential. In natural language, phonological features co-occur with each other. For example, a voiced fricative like [z] might be identified as being [+voice] based on F0 and [+cont] based on aperture in the oral tract. These articulatory events occur more or less at the same time. It is not the case that in the production of [z] there is a period of voicelessness followed by a period of continuancy. On the other hand,
this is what happens in Verhoef et al. (2011): the “features” of a whistle that were identified by the authors are always sequential portions of the signal.

In Verhoef et al. (2013), non-sequential features were reported. For instance, the same pitch shape would be used in two different words, but in one the pitch was smooth and in the other it was broken up into smaller parts. Unfortunately, the authors make no further mention of this, and in their analysis of the entropy of the whistle signals they seem to have only considered sequential portions.

These experiments also avoid discussing the topic of sound change entirely. Change occurs in the experiment when speakers cannot remember how to produce something, and their errorful production is relearned by the next participant. This is quite unlike how sound change occurs in natural language, where it is driven by articulatory or perceptual factors, not memory recall errors.

4.3.5 Hypothesis #3 - Sound change and feature economy

I propose a different approach to this origin of feature economy. I do not think that feature economy functions as any kind of basic organizing principle that all languages are bound to achieve. Instead, I suggest that feature economy is emergent from the way that sound change operates. In particular, it derives from the fact that sound changes affect phonetic properties (“features”), not specific sounds. For example, a change such as final devoicing is one that affects the voicing of obstruents in a particular environment (even if not all obstruents are equally likely to undergo the change). It is not a change that literally turns /b/ into /p/ and /d/ into /t/, etc. Thus, sound change has the effect of creating a new class of sounds that differs by exactly one feature (whichever feature was affected by misperception) from an existing class of sounds, and this tends to lead to an increase in economy.

The key word here is “tends”. Languages could be considerably more economical than they actually are. There are many conceivable, highly economical inventories that are not known to exist. Perfect economy is rarely achieved because of other factors. One such factor is that not all members of a set are equally affected by misperception. For example, Pape et al. (2003) found that word-initial [b] was much less prone to devoicing than word-initial [g], although this also depended on the quality of the following vowel.

Another factor is that some combinations of phonetic features are difficult or impossible. Glottal stops literally cannot be voiced, so if a language has a voicing contrast in its stop series and also [ʔ], then perfect economy is impossible to achieve. It is quite common for languages to have matching sets of voiced and voiceless obstruents, but it is extremely rare to find voicing contrasts in sonorants. This is probably due to articulatory and acoustic-perceptual factors. In languages where voiceless sonorants are reported, phonetic analysis shows that they are at least partially voiced, see e.g. Ladefoged (1995) on voiceless approximants in Tee, and Dantsuji (1984) on voiceless nasals in Burmese. (However, these sounds are considered to be phonologically [−voice], which is what would matter for the purposes
of calculating feature economy.) This means that voiceless sonorants will be unlikely to survive long-term cultural transmission, excluding them from most inventories. The fact that their presence in an inventory would increase economy does not outweigh the difficulties in production and perception.

**Hypothesis #3**

Feature economy effects are emergent from the fact that sound change affects phonetic features, rather than whole sounds. This creates the possibility that a new set of sounds will emerge in an inventory, all of the members of which differ from an older set of sounds by one feature. This in turn creates the appearance of economy in an inventory.

In other words, economy is an outcome of sound change, but not something necessarily favoured by cultural transmission. Inventories with both high and low economy scores are just as likely to be successfully retransmitted. It is inevitable that all inventories will eventually display some kind of economy effects because they all undergo sound change, which targets classes of sounds.

This idea that feature economy is grounded in sound change would help explain the results in Mackie and Mielke (2011). Their randomly generated inventories displayed lower economy values compared to natural language inventories because the randomly-generated ones have not been shaped by millennia of sound change.

### 4.4 Summary

This chapter examined natural language inventories from three different perspectives: the number of sounds, the types of sounds, and the organization of these sounds. In each case, I proposed a hypothesis related to sound change.

In the case of inventory size, the most relevant empirical fact is a correlation between inventory size and phonotactic complexity reported by Maddieson (2007). Languages with more restrictive phonotactics (e.g. languages allowing only CV syllables) tend to have smaller inventories, while languages with more complex phonotactics (e.g. allowing up to CCVCC syllables) tend to have larger inventories. In Hypothesis #1 I proposed that this correlation might actually be causation: since misperceptions are, for the most part, context-sensitive, languages with more complex phonotactics have more diverse phonetic environments and hence a greater variety of sound changes can take place in such a language. Over a long period of time, languages with CCVCC syllables will tend to develop larger inventories than languages with CV syllables simply because a greater variety of sound changes can occur in a CCVCC language.

This chapter also discussed the typological generalization that small and large inventories share a certain number of basic consonants, and as inventories grow they gain more
complex consonants. This was first reported by Lindblom and Maddieson (1988) who examined the inventories of UPSID. In this chapter, I replicated their findings using the inventories of P-base (Mielke 2008). Hypothesis #2 proposed that this could be explained as a result of the interaction between context-free and context-sensitive sound changes. The context-free changes could account for the common “simple” sounds found in nearly all inventories, while the context-sensitive changes could be responsible for the rarer sounds found only in larger inventories (which, by Hypothesis #1 are large due to more complex phonotactics).

Finally, this chapter looked at the issue of feature economy. This refers to the tendency for languages to minimize the ratio between the number of segments in the inventory and the number of feature required to contrast all the segments. Work by Clements (2003, 2009) and Mackie and Mielke (2011) strongly suggests that feature economy is a real property of phonological inventories. Hypothesis #3 proposes that feature economy is the result (but not the goal) of sound change. This is due to the fact that sound change targets broad classes of sounds, and it not specific to individual segments.

These hypotheses rely on some assumptions about sound change that were discussed in Chapter 1. In the following chapter, these hypotheses are tested using simulations from PyILM.
Chapter 5

Simulating inventory evolution

5.1 Introduction

This chapter uses simulations from PyILM to test the hypotheses proposed in the previous chapter regarding cross-linguistic tendencies in inventory structure. In the first set of simulations, I look at how total inventory size grows over time, in particular how the phonotactic patterns of a language affect this growth. The second set of simulations demonstrates how the size principle of Lindblom and Maddieson (1988) can emerge through sound change. The final simulations look at the issue of feature economy (Clements 2003, Mackie and Mielke 2011) and how it changes over time. In all three cases, it will be shown that the observed typological generalizations emerge from cultural transmission, given certain assumptions about sound change.

One important caveat to the results presented here is that the simulation parameters need to be manually tuned. For instance, the similarity threshold of a learning agent is artificially set in order for the simulation to have a “normal” looking outcome. If the threshold is too low, then inventories collapse into a single category almost immediately, if it is set too high then the size of the inventory explodes. The frequency and salience of misperceptions and biases are also somewhat arbitrary. For discussion on how different parameters settings affect a simulation, see Chapter 3. For the complete list of parameters, and their default values, see Chapter 2, Section 2.2.2. Ideally, some of these parameters could be learned by agents, but doing so would be a computational challenge that falls outside the scope of this dissertation.

5.2 Inventory size

In Chapter 4, I provided a hypothesis on inventory size, which I repeat here.
<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>p,b,t,d,k,g,7,m,n,f,s,z, x,l,r,w,j,a,i,u,e,o</td>
</tr>
<tr>
<td>Lexicon size</td>
<td>30</td>
</tr>
<tr>
<td>Generations</td>
<td>30</td>
</tr>
<tr>
<td>Minimum repetitions</td>
<td>2</td>
</tr>
<tr>
<td>Invention rate</td>
<td>0.05</td>
</tr>
<tr>
<td>Min word length</td>
<td>1</td>
</tr>
<tr>
<td>Max word length</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Misperceptions</th>
<th>Target</th>
<th>Feature</th>
<th>Salience</th>
<th>Environment</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop lenition</td>
<td>-voc,cont,-son</td>
<td>cont</td>
<td>0.5</td>
<td>+voc,+voc</td>
<td>0.25</td>
</tr>
<tr>
<td>Nasalization</td>
<td>-cont,-son,-nasal,-voc</td>
<td>nasal</td>
<td>0.5</td>
<td>_+nasal,+son,-voc</td>
<td>0.25</td>
</tr>
<tr>
<td>Initial fortition</td>
<td>-voc,+cont,-nasal</td>
<td>cont</td>
<td>0.5</td>
<td>#</td>
<td>0.25</td>
</tr>
<tr>
<td>Stop aspiration</td>
<td>-voc,-son,-voice,-cont</td>
<td>hisubglpr</td>
<td>0.5</td>
<td>#,+voc,+high</td>
<td>0.25</td>
</tr>
<tr>
<td>Post-nasal fortition</td>
<td>-voc,+cont,-son</td>
<td>cont</td>
<td>-0.5</td>
<td>+nasal,+voc,+son_</td>
<td>0.25</td>
</tr>
<tr>
<td>Labialization</td>
<td>-voc,-son,-cont,-back</td>
<td>round</td>
<td>0.5</td>
<td>+voc,+round</td>
<td>0.25</td>
</tr>
<tr>
<td>Obstructing glottalization</td>
<td>-voc,-son,-cont,-voice</td>
<td>mvglotcl</td>
<td>0.5</td>
<td>-voc,+glotcl,-mvglotcl</td>
<td>0.25</td>
</tr>
<tr>
<td>Retroflexion</td>
<td>-distr,-cont,+cor, -son,-voc</td>
<td>ant</td>
<td>-0.5</td>
<td>_+voc,-ant,-distr</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 5.1: Configuration for testing phonotactic effects on inventory size

**Hypothesis #1** Inventory size is tied to phonotactic complexity, since sound change is partly context-sensitive, and phonotactics defines the set of possible contexts in a language. Languages with more permissive phonotactics should tend to eventually develop larger inventories than those with more restrictive phonotactics.

This hypothesis can be tested through simulation by running multiple simulations with the same starting conditions, varying only the phonotactic restrictions. I begin with a simple illustration of three simulations. Table 5.1 shows the configuration details. Following along the lines of Maddieson (2005), I choose three conditions: the simple phonotactics are maximally CV, the moderate phonotactics are maximally CVC and the most complex permit up to CCVCC. It is important to note that these syllable shapes are the maximum possible, not the only ones possible (see Section 2.2.2.8 on the phonotactics parameter for more information). Furthermore, words in these simulations can have as many as three syllables. Multi-syllabic words can have any combination of possible syllables, e.g. in the CV condition a two syllable word could be any one of the following: CV.CV, CV.V, V.CV, V.V.

The simulations are run with a set of misperceptions based on sound changes collected from Hock (1991) on the assumption that these sounds changes could plausibly be explained as misperceptions, in the sense introduced earlier in this dissertation (see Section 3.2 on page 66). Without proving in each case that they are, I have tried to select sound changes that are indicated to have occurred in numerous languages, and avoid sound changes that are specific to just one language. There are eight misperceptions, listed in Table 5.1.

Let us consider first how the CV language could evolve over the simulation. To begin
with, not all of the misperceptions can possibly have an effect, because not all of the necessary environments exist. Indeed, there are only two possible environments for consonants in such a language: they are either word-initial and prevocalic, or they are intervocalic. Four misperceptions can occur in these environments: stop aspiration and fortition in initial position, labialization and lenition intervocalically.

In contrast, the CVC and CCVCC languages will evolve differently, because it is possible for every misperception listed in Table 5.1 to occur (assuming, of course, that the lexicon contains items in which the appropriate segments occur in the relevant environments). Nonetheless, there are still phonotactically-induced differences in the probability of a misperception occurring. For instance, consider the post-nasal fortition change, which requires the environment [+nasal, −vocalic]. In the CVC language this context only occurs in words with 2 or more syllables (because CC clusters can only happen at syllable boundaries) which reduces the total number of words in the lexicon where this context might occur. Consonant clusters are overall more likely in CCVCC languages because they can potentially occur even in words that are only one syllable long.

It is also worth noting that in CV languages, the number of misperception-triggering contexts is very large, relative to the total number of possible contexts. In fact, 100% of the contexts in which consonants can occur are contexts in which some kind of misperception can happen. In CVC and CCVCC languages, all possible misperception-triggering contexts can occur, but so can a large number of non-triggering contexts.

This is of course partly due to the artificial nature of the simulations, which include only a small number of misperceptions. It would be possible to run simulations with a larger number of misperceptions which cover a larger number of contexts, but the point remains: the number of misperceptions that can occur in a simulation depends largely on the phonotactics.

This has an effect on how the size of inventories changes over time. To illustrate, suppose a language with CV phonotactics has /p/ in the inventory at generation 0. Assuming the misperceptions listed in Table 5.1, all instances of this sound have the potential to become /pʰ/ when they occur word-initially, and to become /f/ when they occur between vowels. The original sound /p/ is not likely to survive an entire simulation. This results in a net gain of one phoneme: the inventory grows from /p/ to /pʰ, f/.

In comparison, in languages with more complex phonotactics, such as CCVC, there are environments in which /p/ can continue to exist, in addition to the environments in which it will undergo change. For instance, it might occur as the first element in a complex onset, or as a coda consonant. In this case, the final generation might still contain words with a /p/ in them. This results in a net gain of two segments, one more than with a CV language: the inventory grows from /p/ to /p, pʰ, f/.

In other words, inventories grow as misperceptions create allophones, and allophones become phonemes. This transition to phoneme, in PyILM, requires one of the following two conditions to be true: (a) the allophone comes to dominate over the older phoneme in a context where misperception occurs; (b) the allophone appears in a new context through
the process of lexical invention.

5.2.1 Simulation results

For a test of Hypothesis #1, I ran 50 simulations under each of the three phonotactic conditions (for a total of 150 simulations). Starting inventory sizes ranged from 10 to 20 consonants which were generated by sampling uniformly at random (with replacement) from the pool of segment symbols in P-base. The results are given in Figure 5.1, which shows the average total size of inventories at each generation over 50 generations of simulation. An ANOVA was performed with final inventory size as the dependent variable and phonotactic condition as the independent variable. The results are significant with $F(2, 147) = 18.287$, $\eta^2 = 0.199$, $p < 8 \times 10^{-8}$.

The expected pattern emerges. The simpler phonotactics resulted in smaller inventories, while the more complex phonotactics resulted in larger inventories. All the languages initially undergo a large increase in inventory size as the various misperceptions take hold. The rises and falls in inventory size occur because of mergers and splits.

![Figure 5.1: Average inventory size for 50 simulations over 50 generations, across 3 different phonotactic conditions.](image)

Inventory size initially grows rather quickly, and this is due to the salience values and probabilities assigned to the misperceptions. If misperceptions had been assigned lower probabilities, or had lower salience values, then the growth in inventory size would be different. The interaction between misperceptions and rate of growth was discussed in section 3.4.

The simulation results, along with the typological correlation reported in Maddieson (2005, see also Section 4.1), strongly support the idea that inventory size is partially deter-
mined by the phonotactics of a language. The effect occurs because of the partly context-sensitive nature of sound change. Phonotactics defines the set of possible contexts in a language, which in turn determines the possible misperceptions that can take place. More complex phonotactics means that a greater variety of misperceptions are possible, which means there is the potential for a greater number of sounds to enter the inventory.

This is not to suggest that phonotactics is the sole factor that determines inventory size. There are clear counter-examples among the Khoe-San languages, which have the world’s largest inventories yet tend to have simple CV syllable structure. Other factors are evidently at play. In particular, it is possible for phonotactic patterns to change over time. This means that language can acquire certain sounds at a stage where they have CVC syllables, and potentially retain those sounds after a later change to a CV syllable structure. The simulations in PyILM make the simplifying assumption that phonotactics are fixed over time, and there are no sound changes that can alter them (e.g. no deletions or epenthesis). See Oudeyer (2005a, 2005b, 2005c) for an example of a simulation of the evolution of phonotactics.

Nonetheless, an explanation for inventory size based on phonotactics is more satisfying than one based on population size (e.g. Atkinson (2011)), because it is grounded in phonologically-relevant effects of production and perception, factors that are far more likely to influence inventory size than just the sheer number of people using a language.

5.3 Common consonants

Sounds are not all equally represented in the inventories of the world. Some sounds are extremely common, such as /p/ or /m/ which are found in nearly every language, while other sounds are more rare, such as /k'/ or /h/. As discussed in Section 4.2, there is a relationship between the size of the inventory, and the types of consonants that an inventory tends to contain. Small inventories generally consist of the most common sounds, while the larger inventories tend to have all the common sounds, plus some more rare ones. Lindblom and Maddieson (1988) (and see also Maddieson (2011)) proposed that this effect is due to how inventories grow over time, by first populating a small space of “neutral” consonants, then gradually make use of more complex sounds.

This superset effect is interesting because it is not an obvious outcome from a misperception-only view of sound change. If the contexts for misperceptions exist in the lexicon, then misperceptions should happen. The overall size of the inventory should not block or facilitate a sound change, and the apparent complexity of the outcome should be irrelevant. How can this distribution of consonants be explained by blind sound change?

The phonotactic effect discussed in the last section plays at least some role. As an example, consider ejectives. These are relatively rare cross-linguistically and they are also on Lindblom and Maddieson’s list of complex sounds, more likely to be found in larger inventories. The emergence of ejectives often requires a sequence of two consonants. For
example, post-nasal stops can develop into ejectives, as occurred in Zulu (Herbert 1985) where Proto-Bantu *p, *t, *k > pʰ, tʰ, kʰ except after nasals where *p, *t, *k > p’, t’, k’. Ohala (1997) argues that ejectives can emerge from a sequence of a plosive and a glottal stop, when the closure for the glottal stop overlaps with closure for the plosive, e.g. the sequence [k] + [ʔ] can result in [k’].

These are contexts that cannot occur at all in a language with simple CV phonotactics, but can occur at syllable boundaries in CVC languages, and even within a single syllable in CCVCC languages. Therefore, ejectives are more likely to emerge in CCVCC or CVC languages, which are also more likely to have larger inventories, by Hypothesis #1.

This only addresses half of the issue, which is the question of why inventories diversify as they grow. It does not explain why small inventories tend to look similar to each other, or what happens as inventories shrink. To address this, it will be necessary to make some modifications to the way that misperception is modeled in PyILM.

5.3.1 Misperception vs. bias

One problem with the current misperception model is that simulations can reach a point where languages cease changing. After running a simulation sufficiently long it will be the case that for any given context in the lexicon, the sounds that appear in that context will either be (a) sounds from the 0th generation of the simulation that are unaffected by any misperception, or (b) sounds that are the result of any misperception that can occur in that context.

This can be visualized as a state diagram, as shown in Figure 5.2. This represents possible states of the inventory in an extremely simplified simulation with only three starting consonants /b, d, g/. Assume there exists a single misperception of final devoicing, and assume these sounds appear in word-final position in the 0th generation. Each circle is a possible inventory, and arrows represent directions of change. The top green circle is the initial state, and the bottom red circle is the only possible final state. A change in state occurs when the devoicing misperception has changed a voiced stop into a voiceless one.

Figure 5.2 is only a partial state diagram, since in an actual simulation run, the inventory can enter some “in-between” states where the voiceless and voiced obstruent both exist together, e.g. /p, b, d, g/. The in-between states inevitably end with the voiceless consonant winning out over the voiced one, so I have excluded these states from the diagram for clarity. Additionally, the state diagram only represents how the stop system in final position evolves. Outside of this context, the assumed misperception does not apply, so at any given state, the inventory will also contain whichever voiced stops are not in final position.

There is a single terminal state that, once reached, cannot be exited. There are no misperceptions that can change a /p, t, k/ inventory back into a /b, d, g/ inventory. Such a state is called an “absorbing state”. If the simulation is run for long enough, the language will eventually reach this state. In this simple example, there is a single misperception, so there is a single absorbing state.
Figure 5.2: State diagram for word-final obstruents in a simulation with final devoicing

In a simulation with a richer set of misperceptions, the state diagram becomes more complicated. More than one absorbing state might exist. Given a large enough set of misperceptions, it may even be technically possible to have a feeding loop. Consider these four changes:

\[
\begin{align*}
A & \rightarrow B / \_ C \\
C & \rightarrow D / B \_ \\
B & \rightarrow A / \_ D \\
D & \rightarrow C / A \_ \\
\end{align*}
\]

When A changes to B before C, it creates the right conditions for C to become D. This in turn creates the right conditions for B to go back to being A. This causes D to turn back into C, and we return to the original state, ready to loop again. However, this requires an extremely specific set of misperceptions and an extremely specific lexicon, and there must be no other misperceptions that could break the loop. Such a set of changes is not likely to arise in natural language, or at least not commonly enough to play any significant role in modeling sound change. That is, we should not use PyILM to construct loops like this because the outcome of such simulations will tell us little about the way that natural languages evolve.

The possibility of absorbing states seems very unnatural. All human languages are constantly undergoing sound change. It would be desirable that languages in a simulation do the same. Of course, at some point language change has to “stop” in a simulation because
simulations are finite. A more realistic goal is to have a simulation where languages at least have the potential to continue changing right until the final generation.

Absorbing states also make it impossible to simulate the right conditions for the phenomenon under discussion in this section. When inventories shrink over time, they must shrink back towards a common set of sounds, otherwise the superset relations found in UPSID and P-base would not exist.

Why do absorbing states occur in a simulation? It is because misperceptions are both context-sensitive and asymmetrical. The probability of $A > B / _C$ is not the same as that of $B > A / _C$. One of those probabilities is usually equal to zero, while the other is non-zero. This pushes languages in a particular direction, without giving any way for the language to return to the state that it used to be in.

One way of addressing this, and allowing inventories to return to former states, would be to make misperceptions symmetrical. For every $A > B$ change, ensure there also exists a $B > A$ change, in the same environment. Symmetrical misperceptions would be very easy to model in PyLIM. A rounding misperception, such as $[\text{round}, \text{voc}] \rightarrow +.5\text{round}$ / \_\_[+round, +voc], could be coupled with an unrounding hypercorrection $[\text{+round}, \text{voc}] \rightarrow -.5\text{round} / \_\_[+\text{round}, +\text{voc}]$. With these misperceptions at play, the inventory of a language can potentially continue to change “forever” (i.e. until the very last generation of a simulation), possibly bouncing back and forth between states.

This may solve the problem of absorbing states, but it lacks empiral support. There are many kinds of sound changes that are clearly not symmetrical. For example, intervocalic voicing of stops has been observed in numerous languages, but the reverse pattern of intervocalic devoicing is vanishingly rare.

Instead of modifying the symmetry of misperception, a better approach is to balance out their context-sensitive nature by introducing context-free changes. As useful terms for discussion, context-sensitive changes will continue to be referred to as misperceptions, and context-free changes will be referred to as biases. The idea behind a bias is essentially the same as a misperception: it is an articulatory or perceptual effect that interferes with the transmission of sounds, and creates the potential for a learner to acquire a different set of sounds than the speaker intended to transmit.

This idea was proposed in the previous chapter as Hypothesis #2, repeated here:

**Hypothesis #2**

Common sounds exist because of context-free biases in transmission that affect all languages, regardless of phonotactics. Rarer segments are rare because they require more specific phonetic environments to appear, and these are more likely to exist in larger inventories, because larger inventories have more, and more different, phonetic contexts (by Hypothesis #1).

Biases, in contrast to misperceptions, are factors that always affect the production or
Table 5.2: Configuration for simulations comparing simple misperceptions and biases

perception of certain classes of speech sounds, regardless of where they occur. For instance, stops are biased towards being voiceless. This is because the conditions for voicing require a certain difference between subglottal and supraglottal air pressures, which is more difficult to maintain when airflow is stopped (Ohala 1983). This of course does not make voicing of stops impossible, it is just more likely that any stop, regardless of where it is produced, could be articulated as voiceless.

This means that, all else being equal, voiced stops are constantly at risk of being misperceived as voiceless, because speakers might, at any point, fail to reach the right air pressure differential for voicing to occur. The fact that certain conditions seem to encourage this even more (e.g. word or utterance final position, Blevins (2006b)) compounds the likelihood of voiceless stops being in any given inventory.

Formally speaking, biases can be modeled in almost exactly the same was as misperceptions within PyILM. They have the same effect of changing the phonetic values of certain sounds, but they can occur in any context, rather than in a specific one. In the notation of PyILM, a context-free misperception is indicated with a * for the environment.

Just as with misperceptions in PyILM, biases are abstractions, and using them in a simulation is not intended to be an argument for the existence of any particular kind of bias in real language transmission. Any proposal for a bias would need to be argued for on its own merits. Here, I will simply be assuming the general existence of biases in order to demonstrate a point: simulated inventories that are both small and large will share sounds (namely, those which emerge through context-free bias), while larger inventories will additionally have rare or more complex sounds (namely, those which emerge through context-sensitive misperception).

To demonstrate, I will first describe two simple simulations, each with a single misperception, and a single bias, so that their interaction is easier to follow. Table 5.2 shows the configuration for this example.

For one simulation, the starting inventory was set to /p, t, k, i, u, a/ and for the other simulation the starting inventory was /b, d, g, i, u, a/. In other words, each of the
simulation conditions started with either a set of stops preferred by bias or the set of stops preferred by misperceptions.

It took several test runs of PyILM to decide on probabilities and salience values for the changes. The context-free nature of biases means that there are more opportunities per utterance for a bias to influence speech than for a misperception to do so. If the bias probability is set too high, it can overpower a misperception, and lead to an absorbing state. The idea, then, is to model biases as frequent but weak effects, while misperceptions are strong but less frequent. In this particular case, the bias is twice a likely as the misperception, but its effect is small enough on a given utterance that it probably will not change which category of sound is understood by the listener. The effect is felt only over many exposures to tokens of a category affected by bias. Misperceptions occur half as often as biases, but their effect is strong enough that a listener will probably perceive a categorically distinct sound.

With the right balance, inventories produced by this kind of simulation will never stop changing, unlike inventories in simulations with only misperceptions. The specific probabilities and salience values for biases and misperceptions will have a strong effect on how frequent and how abrupt the changes are (see Chapter 3, sections 3.2 and 3.3 for more discussion on these parameters). Table 5.3 depicts the consonant inventories for a select number of generations from one simulation. Figure 5.3 shows continual changes in inventory size over 100 generations of two simulations, starting with either voiced or voiceless consonants.

The constant change in inventory size comes from the interplay of the bias and the misperception. Assume an inventory that begins with only /p, t, k/. The misperception will create voiced stops between vowels, which increases the size of the inventory. The bias reduces the voicing in all contexts, so some of these intervocalic voiced stops can merge back with voiceless stops, decreasing the inventory size. The reverse occurs in a simulation that begins with /b, d, g/. Some of these will devoice due to the bias, increasing inventory size, while the intervocalic voicing misperception can cause the voiceless sounds to merge back with the original voiced categories. This can be seen specifically for /g/ in Table 5.3. No /g/ exists in Generation 10. By generation 25, /g/ has emerged as an allophone of /k/. It becomes a full phoneme by Generation 50, then disappears again by Generation 70.

5.3.2 Simulation results

In order to test Hypothesis #2, I ran 120 simulations, with both biases and misperceptions. Some, but not all, balance each other out. For example, I created a labialization misperception, then an anti-labialization bias. These ideas were roughly based on the descriptions of Set 2 and Set 3 consonants in Lindblom and Maddieson (1988). In addition, I created a small number of misperceptions and biases that do not counteract each other. The full set is listed in Table 5.4. As with the Hypothesis #1 test, simulations were divided evenly between CV, CVC, and CCVCC phonotactics (40 each), each with randomly generated
Table 5.3: Example of individual inventories in a simulation with misperception and bias, starting from only voiceless stops

![Table 5.3: Example of individual inventories in a simulation with misperception and bias, starting from only voiceless stops](image)

Figure 5.3: Change in inventory size for two simulations, one starting with voiceless stops, one with voiced stops

![Figure 5.3: Change in inventory size for two simulations, one starting with voiceless stops, one with voiced stops](image)
starting inventories, consisting of anywhere from 8-12 consonants, and 3-5 vowels, selected at random from P-base.

Each simulation ran for 50 generations, and the final inventory of each was collected. The expectation is that context-free biases will be responsible for a set of sounds found in most inventories, while the context-sensitive misperceptions are what lead to more rare sounds in larger inventories.

The segments of the final inventories were therefore roughly categorized this way: sounds that are the possible outcome of a bias were counted separately from the others. For instance, there is a bias against retroflex stop consonants. Retroflex is represented as [+ant, –distr, –cont, +con, –son, –voc] in PyILM. The bias affects anything with that feature set, and raises the [ant] value, thus a segment marked [+ant, –distr, –cont,...] is a possible outcome of a bias, and would be flagged.

The initial inventories of the simulation were generated by sampling uniformly at random from the set of all possible “biased” and “other” (non-biased) sounds. The total number of biased segments is much smaller than the total number of other segments and so the initial inventories tended to have a high proportion of non-biased sounds. If sound change had no effect on the relative proportion of segment types, then the expectation would be for the inventories of the final generation to also have a greater proportion of non-biased sounds. As Figure 5.4 shows, however, this is not what happens. Instead in smaller inventories, the number of biased segments is sometimes equal to or greater than the other segments.

Table 5.4: Misperceptions and biases for testing Hypothesis #2

<table>
<thead>
<tr>
<th>Biases</th>
<th>Target</th>
<th>Feature</th>
<th>Salience</th>
<th>Environment</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sonorants are voiced</td>
<td>son</td>
<td>voice</td>
<td>0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
<tr>
<td>Nasals are voiceless</td>
<td>nasal, non-son</td>
<td>voice</td>
<td>0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
<tr>
<td>Nasals are sonorant</td>
<td>nasal</td>
<td>son</td>
<td>0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
<tr>
<td>Obstruents are voiced</td>
<td>+son, –nasal</td>
<td>voice</td>
<td>0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
<tr>
<td>Anti-sonorant bias</td>
<td>+son, –nasal</td>
<td>high</td>
<td>-0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
<tr>
<td>Anti-aspiration bias</td>
<td>+son, –nasal</td>
<td>+subgl</td>
<td>+subgl</td>
<td>-0.1</td>
<td>+</td>
</tr>
<tr>
<td>Anti-retroflex bias</td>
<td>+cont, –nasal</td>
<td>+cor, +voc</td>
<td>mvgl, cl &lt; +mvgl, cl &gt;</td>
<td>+0.1</td>
<td>+</td>
</tr>
<tr>
<td>Anti-labilization bias</td>
<td>+son, –nasal</td>
<td>round</td>
<td>-0.1</td>
<td>+</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Misperceptions</th>
<th>Target</th>
<th>Feature</th>
<th>Salience</th>
<th>Environment</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larynx</td>
<td>+voc, +cont, +con</td>
<td>cont</td>
<td>0.3</td>
<td>+voc, +voc</td>
<td>0.45</td>
</tr>
<tr>
<td>Nasalization</td>
<td>+cont, +nasal, +voc</td>
<td>nasal</td>
<td>0.5</td>
<td>+nasal, +voc, –son, +voc</td>
<td>0.25</td>
</tr>
<tr>
<td>Nasal fortson</td>
<td>+con, +cont, –nasal</td>
<td>nasal</td>
<td>-0.5</td>
<td>+nasal, –voc, +son</td>
<td>0.25</td>
</tr>
<tr>
<td>Post-nasal fortson</td>
<td>+voc, +cont, +son</td>
<td>cont</td>
<td>-0.5</td>
<td>+nasal, –voc, +son</td>
<td>0.25</td>
</tr>
<tr>
<td>Labialization</td>
<td>+voc, +cont, +back</td>
<td>round</td>
<td>0.5</td>
<td>+voc, +voc</td>
<td>0.45</td>
</tr>
<tr>
<td>Retroflexion</td>
<td>+voc, –nasal, +cor, +son, +voc</td>
<td>cont</td>
<td>-0.5</td>
<td>+voc, +ant, +cor</td>
<td>0.25</td>
</tr>
<tr>
<td>Obstruent glottalization</td>
<td>+voc, +cont, +voc</td>
<td>+voc</td>
<td>0.5</td>
<td>+voc, +voc</td>
<td>0.25</td>
</tr>
<tr>
<td>Stop aspiration</td>
<td>+voc, +voc, +cont</td>
<td>+subgl</td>
<td>0.5</td>
<td>+voc, +ant, +cor</td>
<td>0.25</td>
</tr>
<tr>
<td>Nasal backing</td>
<td>+voc, +nasal</td>
<td>high</td>
<td>0.5</td>
<td>+voc, +high, +low</td>
<td>0.25</td>
</tr>
<tr>
<td>Nasal click voicing</td>
<td>+voc, +nasal</td>
<td>voice</td>
<td>0.5</td>
<td>+voc, +voc</td>
<td>0.25</td>
</tr>
<tr>
<td>Consonant place assimilation</td>
<td>+con</td>
<td>+cor</td>
<td>0.5</td>
<td>+voc, +vo</td>
<td>0.25</td>
</tr>
<tr>
<td>Affrication</td>
<td>+voc, +cont, +son</td>
<td>+cor</td>
<td>0.5</td>
<td>+voc, +vo</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Figure 5.4: Biased and non-biased sounds in the final simulated inventories.

This parallels the relationship found in natural language inventories. Lindblom and Maddieson (1988) used the metaphor of a magnet and a rubber band to explain this. These simulations suggest that the metaphor can replaced by more concrete notions, namely that context-free biases are the rubber bands drawing languages toward common sounds, and the context-sensitive biases are the magnets, pushing inventories to expand into different regions of phonetic space.

5.4 Feature economy

This final section of the chapter turns to the topic of feature economy (Clements 2003, Hall 2007, Mackie and Mielke 2011, see also discussion in section 4.3.2). Economy can be calculated in several ways, but it is essentially a measurement of how many phonemes exist in an inventory, relative to the number of phonological features required to keep all phonemes distinct. A discussion of various economy metrics was presented in Section 4.3.2.1. Natural language inventories have been shown to be more economical than randomly generated sets of segments (Mackie and Mielke (2011)). In the previous chapter, I introduced a hypothesis about the diachronic origin of feature economy, which will be tested through simulation in this section. I repeat the hypothesis here:

**Hypothesis #3**

Feature economy effects are emergent from the fact that sound change affects phonetic features, rather than whole sounds. This creates the possibility that a new set of sounds
will emerge in an inventory, all of the members of which differ from an older set of sounds by one feature. This in turn creates the appearance of economy in an inventory.

This hypothesis does not predict economical inventories to be favoured by cultural transmission. Greater economy does not mean greater learnability. Misperceptions and biases are a more powerful force. For instance, it is common to find voicing contrasts among obstruents, but this is rare among sonorants, even though it would be much more economical to use the [voice] feature across the entire consonant inventory. The articulatory and acoustic-perceptual difficulties associated with voiceless sonorants outweigh any increase in economy that might result from adding them to the inventory. In other words, my claim is that economy emerges as a side-effect of sound change and since all natural languages undergo sound change, all of them display some degree of economy.

This section is organized as follows. First, I will describe the general relationship between sound change and economy, and provide a simple simulation as illustration. Next, I will move to testing the hypothesis more directly. This will be done by comparing results of simulations run with misperceptions that affect classes of sounds, and simulations run with misperceptions targeting individual sounds. If the hypothesis is correct, then economy will be ultimately higher in cases where misperceptions are defined over classes.

5.4.1 How economy can change over time

Economy scores are all calculated using two values: $S$, the number of segments in the inventory, and $F$, the minimum number of features needed to contrast the inventory. Economy changes as $S$ and $F$ change. Economy scores are raised if either $S$ increases without increasing $F$ or else $F$ decreases without decreasing $S$. Increasing both $S$ and $F$ an equal amount will result in a lower economy value, while decreasing both $S$ and $F$ and equal amount will result in an increase in economy. This is true regardless of which of the four economy metrics are used. Let us consider in turn how each of these values can change.

$S$ changes whenever a segment is added or lost. Sounds are added to the inventory through phonemic split, when misperception changes an existing sound in a particular context. In most cases, splits result in an increase to $S$. If a sound happens to have its (lexical) distribution strictly limited to contexts affected by a misperception, then the new sound completely replaces the old one, and $S$ does not change. It is possible for $S$ to decrease due to merger, when all instances of one category become instances of another existing category.

Note that sound change can occur without any change to $S$ at all. For instance, suppose that an inventory consists of /b, t, d/, and all instances of /b/ devoice to /p/. This is not a merger, since /p/ was not already in the inventory, nor is it a split, because there are no instances of /b/ left behind. The resulting inventory /p, t, d/ is the same size as the original inventory, so $S$ is unchanged.

How about changes to $F$? They come along with changes to the inventory. As far as the simulation is concerned, it would be impossible for $F$ to change without a change in
inventory, since $F$ is determined using the Feature Economist algorithm (see section 4.3.3). The same input inventory will always result in the same number of output features. $F$ increases if a segment is added to the inventory that requires a feature which previously was not necessary. For instance, if an inventory has only stops contrasting in voicing and place, and a fricative is added, then $F$ will increase as [continuant] becomes a necessary feature.

Losing a segment may result in a decrease of $F$ if this segment was in minimal contrast with another segment that now stands alone. For instance, imagine an inventory with a series of voiceless obstruents, only one of which has a voiced counterpart, e.g. /p, t, k, s, z/. The feature [voice] is necessary to contrast /s, z/ only. If /z/ were to disappear, then the need for [voice] as a feature is also lost, and so $F$ and $S$ both decrease (and the same would be true if /s/ were lost instead).

There is an effect that occasionally occurs in PyILM simulations, call it “feature carry-over”, which can result in a decrease in feature economy. Suppose there is an inventory with six segments /p, b, ṯ, d, f, s/. The feature economy of this inventory is relatively high, as only three features are required to contrast everything: [voice, continuant, coronal]. The palatalization on /ṯ/ obviously involves another feature too, and realistically, a speaker of such a language would need to have this additional articulatory information represented somehow. However, for the purpose of simply working out the smallest number of phonological features necessary to contrast these segments, only three are required.

Suppose there is a sound change that increases the [continuant] value of a segment, e.g. stops become fricatives, and that through this sound change the /ṯ/ lenites to /s̱/. Now, it is necessary to introduce a feature for palatalization in order to contrast /s/ and /s̱/. This has an overall effect of lowering economy, because although $S$ increases by 1, $F$ also increases by 1 (this is assuming that some instances of /ṯ/ do not undergo change; if every /ṯ/ becomes /s̱/ then $S$ would not increase, but $F$ would, and economy would decrease even more). Changes that increase both $S$ and $F$ by the same amount will result in an overall decrease in economy. This is simply due to the fact that for any inventory $S > F$, therefore an increase in $S$ is proportionally less than an equal increase in $F$. Consider a series of sound changes that each add one segment and one feature. Using the Simple Ratio measurement of economy, we would get this series of shrinking values: $3/2 = 1.5$, $4/3 = 1.333\ldots$, $5/4 = 1.25$, $6/5 = 1.2$, and so on.

The simple ratio economy of /p, b, ṯ, d, f, s/ is $E = 6/3 = 2$ whereas adding in /s̱/ means $E = 7/4 = 1.75$. This effect of feature carry-over can occur in natural language changes, for instance a rounded back vowel may front, carrying its roundedness with it, potentially creating a contrast with an existing unrounded front vowel.

Change in economy over time is illustrated in Figure 5.5 for a hypothetical simulation. That is, all the values are constructed for the purposes of illustrating how change in economy happens. No actual simulations were run to obtain these numbers. The top of the figure shows change in Simple Ratio, while the other three metrics (Frugality, Exploitation, and Relative Efficiency) are shown at the bottom. This is because the latter three metrics are
bounded between 0 and 1, while Simple Ratio has no upper limit, so they cannot easily be shown on the same scale.

Each step along the x-axis is comparable to a generation in a simulation. At a glance, one can spot that the metrics do not increase or decrease in a uniform way over time. Consider just the change from the first to the second generation. In the first generation, the inventory is quite small, with only 5 segments and 3 features. At the second generation, both the number of segments and the number of features has grown. This results in Simple Ratio and Frugality both increasing, while Exploitation drops, and nothing happens to Relative Efficiency. These differences make sense, when we consider what each metric is actually measuring.

Simple Ratio, as the name implies, is simply the ratio of segments to features. Since $9/4 > 5/3$, Simple Ratio goes up in the second generation. In fact, Simple Ratio increases steadily until the 8th generation because each subsequent ratio is higher. In the eight generation, Simple Ratio falls because although the inventory size grew, it was not enough to make up for the change in features. The inventory at the 9th generation would need to have 38 segments in order to see a gain in Simple Ratio compared to the previous generation.

Frugality is a measurement of how close an inventory comes to having the minimum number of features it could have, given its size. For $S$ segments, this minimum number is $\log_2 S$, rounded up to the next whole number. Frugality barely changes between the first and second generation because in both cases this minimum number is achieved. For 5 segments, at minimum 3 features are needed, and for 9 segments at minimum 4 features are needed. Frugality is not a strict measure of this ratio however, since these inventories would score 1.0 otherwise. Larger inventories actually receive slightly higher Frugality scores. This can be observed by the way that Frugality changes between the second and third generation. In
both cases, the inventory requires 4 features and in both cases this is the minimum possible for an inventory of that size. These 4 features could potentially be used to contrast as many as 16 segments, and so the third generation inventory with 15 segments scores higher than the second generation inventory with only 9 segments.

Exploitation is, in a sense, the opposite of Frugality, and it measures how close an inventory comes to having the maximum number of segments for a given number of features. The maximum inventory size for $F$ binary features is $2^F$, so for an inventory to remain highly economical on the Exploitation metric, its size must increase by a power of 2 every time a feature is added. In the first generation, with 3 features, it is possible to have as many as 8 segments. The final generation shown in Figure 5.5 has 8 features, so the inventory would by then need to have 256 segments to reach a perfect Exploitation score. This means that generally speaking, sound changes that result in an increase in the number of features will tend to result in a decrease in Exploitation scores. There are only two examples of Exploitation increasing in Figure 5.5: in the third generation when the number of segments increases without any change to the features, and in the eight generation when the number of segments and the number of features both decrease.

Relative Efficiency looks at the minimum and maximum number of features required for a given inventory size, and assigns a score based on where an inventory falls in that range. The first four generation all score perfectly on Relative Efficiency because they all have the minimum possible features. This should be contrasted with Frugality, which assigned different scores to these inventories based on their size. Relative Efficiency falls in the fifth through seventh generations because the number of features rises to 5, while the inventory size stays in a range that could potentially require as few as 4 features. When the number of features drops in the ninth generation, Relative Efficiency once again goes up.

Another issue to consider is that not only does economy change over time, the range of possible scores changes as well. To understand why this is so, it is helpful to plot all possible economy scores for a range of features and segments. This is shown in Figure 5.6 for Simple Ratio and Figure 5.7 for Frugality. The figures show inventory size and the number of features on the x and y axes, while the z-axis shows the feature economy score that a language would have with that combination of segments and features. Not every point in space is filled, because it is not possible for certain combinations to occur. An inventory with $S$ segments needs at minimum $\log_2 S$ features (rounded up).

One important difference to note between Frugality and Simple Ratio is where the minimum scores lie for each feature value. For Simple Ratio, the minimum is the same. An inventory of $S$ segments cannot possibly need more than $S-1$ features, so the minimum Simple Ratio lies just above 1.0 for all values of $F$.

For Frugality, the lowest possible score actually varies with the number of features. If an inventory requires two features, then Frugality cannot go below 0.79. If an inventory requires three features, then the minimum score is now 0.6. At four features the minimum score drops to 0.58, and so on.

This means that sound changes requiring the addition of a new feature to the inventory
Figure 5.6: Range of possible Simple Ratio scores

Figure 5.7: Range of possible Frugality scores
have a greater impact on the Frugality score than Simple Ratio. For example, an inventory with 25 segments and 5 features has a Simple Ratio score of 5.0 and a Frugality score of 0.928. If a sound change occurred increasing the inventory to 26 segments, but at the cost of adding a 6th feature, then both scores will drop. If the inventory can add just four more segments, for a total of 30, then it would regain its old Simple Ratio score of 5. In the case of Frugality, the inventory would need to balloon to 48 segments to equal its old score.

### 5.4.2 An illustrative example

In this section, change in economy scores is illustrated using a simple simulation. The initial inventory was selected to be a simple set of consonants, mainly obstruents, with just three vowels: /b,d,g,q,f,z,x,m,n,i,u,a/. There were four context-sensitive misperceptions included for this simulation, and they only affect the obstruents in the language:

- **Devoicing** [+voice, −son, −cont] segments have their [voice] value reduced by .5 in the environment of _# (p=.25)
- **Lenition** [−son, −cont] segments have their [cont] value increased by .5 in the environment of +voc_+voc (p=.25)
- **Fortition** [−son, +cont] segments have their [cont] value reduced by .5 in the environment of #_ (p=.25)
- **Assimilation** [−son, −voc, −voice] segments have their [voice] value increased by .5 in the environment of +voice, −voc_ (p=.25)

Every sound change was assigned a .25 probability, meaning that on any utterance of a sound in an appropriate context, there is .25 probability that a given misperception occurs. Each misperception alters a token’s feature values by .5, which is a large enough change in value, given the parameters of the learning algorithm, to practically ensure that the tokens affected by sound change will be categorized as something different than tokens of the same segments that go unaffected by change.

The phonotactics of the language are set to be maximally CVCC, and words can be one or two syllables long. The phonotactics are such that it is possible for all four misperceptions to occur at some point. Devoicing can occur in any C-final word, Assimilation can occur in any C-final word. Lenition requires two vowels, which means that it requires a two-syllable word to be triggered. Fortition could happen in any word that starts with a fricative.

At the end of the simulation, the feature economy of the language was calculated at each generation, using the four metrics described in Figure 4.3.2.1. Feature economy, as originally defined by Clements (2003), measured the organization of the phonological inventory, not the full surface inventory. Similarly in simulations with PyILM, only the core or underlying inventory is considered. Allophones, that is sounds which occur uniquely as variants of
others and never on their own, are not counted. See section 3.2 in Chapter 2 for discussion of how allophones are identified in PyILM.

Change in feature economy for this simulation is shown in Figure 5.8. Economy rises gradually over the course of the simulation, reaching a maximum of around 3.6 on the Simple Ratio measurement. Growth is not consistent. There are a few areas where economy hits a plateau, and there are also times when it goes down.

![Figure 5.8: Change in feature economy for a simple simulation](image)

Economy continues to rise, on all metrics, until just past the 20th generation. The initial increase is due to pairs of segments emerging from sound change. Generation 17 represents the peak of the Simple Ratio scores, at 3.6. From here, economy dips and rises, but never falls lower than 3.2.

The reason that economy never settles at a particular score is because some segments are in environments where they are subject to more than one misperception. For example, some voiced stops appear in final position following another voiced stop. This means that both devoicing and voicing assimilation can both apply. A voiceless sound can therefore appear through one misperception, only to be wiped out by the other.

Relative Efficiency evolves differently from all of the others. In some cases, it stays at 1.0, a perfect score, while other scores drop (especially around the 20th generation). This is because, for a given feature value, there is range of segment inventory sizes that will always get a Relative Efficiency score of 1.0. Specifically, for an inventory of size $S$ requiring $F$ features for contrast, Relative Efficiency is 1.0 if it is the case that $2^{F-1} + 1 \leq S \leq 2^F$. For example, an inventory of 28 segments that requires 5 features for contrast will get a perfect Relative Efficiency score because $2^4 + 1 < 28 < 2^5$. In fact, any inventory between 17 and 32 segments will receive the same score.

Practically speaking, this means that an inventory can shrink in size without affecting the Relative Efficiency score, so long as the number of features does not change. This is
just what happens in the simulation shortly after generation 20. The inventory had evolved a voicing distinction at every place of articulation for both stops and fricatives. There were 18 total phonemes (16 obstruents and 2 nasals), and the inventory required 5 features for contrast. That is the minimum possible so Relative Efficiency was 1.0. At a later generation, one of the voicing contrasts collapsed as a voiceless sound merged with a voiced sound. This reduced the inventory size to 17, but without any change to the contrastive features. This caused Simple Ratio to drop, because $\frac{17}{5} < \frac{18}{6}$, but Relative Efficiency was unaffected because $2^4 + 1 < 18 < 2^5$.

5.4.3 Segment-specific misperceptions vs. class-level misperceptions

Hypothesis #3 proposes that feature economy emerges over time because sound changes affect phonetic features, and hence classes of sounds, rather than targeting individual sounds. Importantly, the claim here is not that sound change affects classes of sounds simultaneously. At any given generation, any number of members of a class might be affected, or perhaps none are, but the net result, over many generations, is that a class of sounds will have undergone sound change. In other words, given enough time, sound change can create a new class of segments out of an old one, with the two classes differing by whatever feature was affected in the sound change. This results in inventories with classes of sounds contrasting along particular feature dimensions, i.e. feature economy.

To test this hypothesis, it is necessary to run simulations under two different conditions. In one condition, changes are defined over broad classes of sounds (the “class-level” condition), i.e. the familiar kinds of misperceptions and biases already used in this dissertation. In the second condition, changes target individual sounds in an inventory (the “segment-specific” condition). This is in a sense like simulating two possible worlds, where sound change operates in different ways. The class-level condition represents the actual world, while the segment-specific condition represents a hypothetical alternative world that we can compare against. If Hypothesis #3 is correct, then inventories in the class-level condition should generally have higher economy scores than those in segment-specific condition.

It is important to distinguish between segment-specific changes fabricated for these simulations, and multiple instances of class-level changes affecting individual segments. Even though it is fairly clear that real sound changes affect broad classes of sounds, it does not mean that every sound in a class is equally likely to be affected. Consider the case of a lenition-type change, which increases the continuancy value of a stop between vowels. Hualde et al. (2011) discuss how lenition of intervocalic stops in Romance languages varies with context, and between types of stops. From these findings, we might decide to implement several “segment-specific” lenition misperceptions, one for every plosive in an inventory. Each misperception would have a different salience and probability, and perhaps even somewhat different environments. That way, /b/ would be affected by lenition in its own way, slightly different from how /d/ or /g/ might be affected.

However, this would not be the right approach. All of these lenition changes have
identical outcomes, even if their triggering conditions are different. It would not matter whether we define lenition for each segment, or whether we define lenition over a class of segments, because after a simulation has been running long enough, all stops between vowels will have become fricatives (assuming there is no counter-acting bias). In other words, these are not really segment-specific changes. They are instances of the same class-level change, with minor variations.

Instead, to make something truly segment-specific, then not only would each misperception have to be defined separately over each individual segment, but the outcome of each misperception needs to be different as well. For example, all stops in intervocalic position can still be subject to change, but each stop would have a different feature affected: /b/ might have its [nasal] value increased, and change into /m/ while /d/ might devolve to /t/. In this way sound change is actually targeting individual segments, and not (natural) classes. To be clear, such changes are extremely unnatural, and they are only being introduced as a way of testing Hypothesis #3.

In order to generate these kinds of misperceptions, I made a modification to PyILM. The simulations in the segment-specific condition are initialized with the same set of class-level misperceptions used in the other simulations. At the beginning of each generation, PyILM looks through the inventory of the speaking agent, and checks to see if any of the class-level misperceptions could potentially apply. A misperception is considered to potentially apply if, for any segment, the set of features targeted by the misperception is a subset of the segment’s full feature specification.

If a class-level misperception would apply, then PyILM generates a unique segment-specific one. The new misperception will have the same environment as the class-level one, and it will have the same probability of occurring, but it targets a set of features equal to the full specification of the segment in question. The outcome of the segment-specific misperception has the same salience value as the class-level one, but it applies to a new, randomly-selected, feature. After checking the entire inventory, the simulation runs as normal, but using these segment-specific misperceptions instead of the class-level ones.

At the beginning of each generation, the old set of segment-specific misperceptions is deleted, and a new set is created. This is done to ensure that class-level and segment-specific misperceptions have the same chances of applying in a given lexicon. Due to the highly-restricted nature of segment-specific changes, it is generally going to be the case that they only apply once. Suppose there there is a /b/-specific misperception that changes /b/ to /m/. Once this occurs, that /b/-specific misperception will never do anything else in the simulation, unless by sheer chance another segment-specific change has created a /b/ from something else. In practice, this means that inventories will change only during the first few generations, and then never again. By creating new segment-specific misperceptions at each generation, based on how class-level misperceptions would apply, the outcomes of simulations in either condition should be comparable.

For example, suppose that a simulation has a starting inventory of /p, t, k, z, i, a/, and the following misperceptions.
Devoicing \([+\text{voice}, -\text{son}] \rightarrow [-.5\text{voice}] / \#\), \(p = .25\)

Lenition \([-\text{son}, -\text{cont}] \rightarrow [+ .5\text{cont}] / +\text{voc} + \text{voc}, p = .25\)

The Devoicing misperception could apply to \(\text{/z/}\) and the Lenition misperception can apply to \(\text{/p, t, k/}\) (whether they actually apply depends, of course, on the lexicon having the right environments). In this case, PyILM would create one new misperception for each of \(\text{/z/}, \text{/p/}, \text{/t/}\) and \(\text{/k/}\). The segment-specific misperceptions could look something like this:

**z-Change** \([-\text{voc}, -\text{son}, +\text{cont}, +\text{voice}, -\text{nasal}, +\text{cor}, -\text{ant}, -\text{strid}, -\text{lat}, -\text{back}, -\text{low}, -\text{high}, -\text{round}, -\text{distr}, -\text{glot_cl}, -\text{hi_subgl_pr}] \rightarrow [-.5\text{back}] / \#\), \(p = .25\)

**p-Change** \([-\text{voc}, -\text{son}, -\text{cont}, -\text{voice}, -\text{nasal}, -\text{cor}, +\text{ant}, -\text{strid}, -\text{back}, -\text{low}, -\text{high}, -\text{round}, +\text{distr}, -\text{glot_cl}, -\text{hi_subgl_pr}] \rightarrow [+ .5\text{nasal}] / +\text{voc} + \text{voc}, p = .25\)

**t-Change** \([-\text{voc}, -\text{son}, -\text{cont}, -\text{voice}, -\text{nasal}, +\text{cor}, -\text{ant}, -\text{strid}, -\text{lat}, -\text{back}, -\text{low}, -\text{high}, -\text{round}, -\text{distr}, -\text{glot_cl}, -\text{hi_subgl_pr}] \rightarrow [+ .5\text{round}] / +\text{voc} + \text{voc}, p = .25\)

**k-Change** \([-\text{voc}, -\text{son}, -\text{cont}, -\text{voice}, -\text{nasal}, -\text{cor}, -\text{ant}, -\text{strid}, -\text{lat}, +\text{back}, -\text{low}, -\text{high}, -\text{round}, -\text{distr}, -\text{glot_cl}, -\text{hi_subgl_pr}] \rightarrow [+ .5\text{cont}] / +\text{voc} + \text{voc}, p = .25\)

These misperceptions have the same environment and probability as the original Devoicing and Lenition, but they target a set of features that are specific to one particular segment. They all have the same salience as Devoicing and Lenition, altering a feature value by \(\pm .5\), but which feature they affect is different in each case. As mentioned above, the affected feature is randomly selected.

### 5.4.4 Calculating feature economy

After running a simulation, the feature economy of the inventory at each generation was calculated using the Feature Economist algorithm. The results are reported in the next section, and this section describes the details of how the algorithm works. In brief, it takes a set of segments with full feature specifications as input, and it returns the smallest number of features necessary to contrast every member of that set. The algorithm has been described in Mackie and Mielke (2011), and a version of it appears in P-base (Mielke 2008). For this dissertation I wrote my own implementation.

Calculating feature economy requires two numbers: \(S\), the number of segments in the inventory \(I\), and \(F\), the smallest number of features from a feature set \(\varphi\) required to contrast all of the segments in \(I\). For the purposes of calculating feature economy, two segments are contrastive with respect to a feature set \(\varphi\) if they differ by at least one feature in \(\varphi\). The most useful mathematical tool for this is the concept of a “combination”. A \(k\)-combination of a set \(A\) is an unordered subset consisting of \(k\) elements of \(A\). The problem of finding
the smallest number of features from \( \varphi \) that are necessary to contrast the segments in \( I \) becomes the problem of finding the largest \( k \)-combination of features that are unnecessary for contrast.

The number of \( k \)-combinations that can be drawn from a set of size \( N \) is written as \( \binom{N}{k} \), which is read as “\( N \) choose \( k \)”. This is equal to \( \frac{N!}{k!(N-k)!} \) if \( N > k \), otherwise it is equal to 0.

The Feature Economist algorithm begins with a pre-processing step where non-contrastive features are removed from \( \varphi \). These are features for which every segment in the inventory shares a value. For instance, if there are no laterals in the inventory, then every segment will be \([-\text{lateral}]\). This means there are no contrasts based on \([\text{lateral}]\) so this feature can be discarded immediately.

The algorithm then goes through a loop of creating larger and larger \( k \)-combinations of features. For each \( k \)-combination, the algorithm removes that combination from \( \varphi \), creating a new subset \( \varphi' \). A pairwise comparison of the segments in \( I \) is done to check if each pair is contrastive with respect to \( \varphi' \). If not, that is if two segments become identical without this particular \( k \)-combination of features, then the combination is added to a special list of \text{CRUCIAL} features. If contrast is still possible without this \( k \)-combination, then the set \( \varphi' \) is designated the \text{FINAL} set (replacing any previous \text{FINAL} set), and the algorithm carries on to the next \( k \)-combination. When all \( k \)-combinations have been tried for some value of \( k \), then \( k \) is increased by 1, and the process of removing \( k \)-combinations repeats.

If at any time the size of the \text{FINAL} set and the value of \( k \) differ by 2, e.g. \( k = |\text{final}|+2 \), then the algorithm terminates. It terminates at this point because this difference means that the algorithm has attempted to remove every single \( k \)-combination for some value of \( k \) and did not succeed. Whenever some \( k \)-combination can be removed and contrast is maintained, the contents of the \text{FINAL} set are updated, and its cardinality becomes equal to \(|\varphi'|-k\) (the full set of features, minus the combination currently removed). After trying all \( k \)-combinations, the value of \( k \) is increased by one. If no \( k \)-combinations can be removed for this new value of \( k \), then the contents of the \text{FINAL} list will not change, and \( k \) will again increase. At this point \( k = |\text{final}|+2 \) and the algorithm should halt.

For each value of \( k \), all possible \( k \)-combinations are generated. If the combination is found to be a superset of any element of the \text{CRUCIAL} list, then it is skipped and the inventory is not checked for contrast. For example, if it was previously found that removing \([\text{nasal},\text{son}]\) left two segments identical, then there is no point in trying to remove \([\text{nasal},\text{son},\text{voice}]\). At first, checking every combination slows down the algorithm, and often it needs to do a pairwise comparison of the inventory for every \( k \)-combination up to \( k=4 \). Around this point, the \text{CRUCIAL} list begins to fill in, and after this the algorithm runs much faster as it can immediately reject \( k \)-combinations, without the need to run the pairwise comparison across the inventory.

To help understand some of the numbers involved, here is an example of the algorithm at work. The consonant inventory \([\text{\textipa{[i]}}, \text{\textipa{[h]}}, \text{\textipa{b}}, \text{\textipa{n}}, \text{\textipa{k}p}, \text{\textipa{\textipa{d}}}}, \text{\textipa{\textipa{j}}}, \text{\textipa{\textipa{p}n}}, \text{\textipa{kw}}, \text{\textipa{fi}}, \text{\textipa{f}}, \text{\textipa{n}}, \text{\textipa{r}}\] was randomly generated and given as input. The initial set of features consisted of 19 features.
The pre-processing step removed 2 non-contrastive features. Early in the calculation, most of the feature possibilities need to be considered. At $k=5$, for example, the number of 5-combinations is $\binom{17}{5} = 6,188$ and the algorithm tried a pairwise comparison of the inventory for 3,374 of those combinations (87%). At the point where $k=10$, the number of 10-combinations is $\binom{17}{10} = 19,448$ but the algorithm only checked the inventory for contrast using 3,165 of them (16%), because the rest were supersets of combinations that failed earlier. This saves more than 10,000 comparisons at this step. Eventually, the algorithm removed 12 more features, thus a minimum of 5 features are needed to contrast this inventory.

5.4.5 Simulation results

For each of the class-level and segment-specific conditions, I ran 90 simulations, for a total of 180 simulations. Each condition was broken into three phonotactic groups: 30 simulations with CV phonotactics, 30 with CVC phonotactics, and 30 with CCVCC phonotactics. This ensures that a variety of different inventories will emerge, both in term of inventory size and contents.

Starting inventories sizes varied as well, and each phonotactic group was broken in three size categories: 10 simulations started with small consonant inventories, ranging from 8-15 consonants, 10 started with medium-sized inventories between 20-40 consonants, and 10 started with large inventories of 60-80 consonants. Every inventory had between 3 and 5 vowels, although no misperceptions affect vowels so this set remained constant for the entire simulation. Each simulation ran for 30 generations.

The starting inventories were generated by sampling uniformly at random from P-base. Only 90 inventories were created, and these were shared across the two conditions. Additionally, the 90 starting lexicons from the class-level condition were re-used in the segment-specific condition. In other words, for each class-level simulation, there was a segment-specific simulation that had identical starting conditions.

Figure 5.9 shows change in average feature economy scores for the 90 simulations with class-level changes, for all four metrics. Results for simulations with segment-specific misperceptions are given in Figure 5.10.

Overall, there is an increase in economy on the Simple Ratio metric, in both types of simulations. The Exploitation metric, on the other hand, goes down in both. Changes in Simple Ratio and Exploitation can mostly be explained by changes in inventory size. Inventories generally grow in size over the course of a simulation. Larger inventories tend to score higher on Simple Ratio, and they tend to score lower on Exploitation. This relationship between size and economy score was discussed in more detail in Section 4.3.2.1.

To see if the differences between the two simulation types were significant, I fit the economy scores from generation 30 to a linear regression model. The independent variables were inventory size and misperception type (class-level or segment-specific). The dependent variable was economy score. It is important to note that my choice to use the economy
Figure 5.9: Change in average feature economy for simulations run with class-level changes

Figure 5.10: Change in average feature economy for simulations run with segment-specific changes
Table 5.5: Results of two-way ANOVA with inventory size and misperception type as predictors and economy score as dependent variable.

Scores from generation 30 is somewhat arbitrary. It is not clear that the economy scores have completely stabilized at this point, nor is it clear how many generations would be sufficient. Ideally, one would calculate a stationary distribution of inventories, but this is challenging given the extremely large space of possible inventories that could evolve in a simulation.

The model was calculated using the \texttt{anova} function from the R programming language (R Core Team 2016). Results are shown in Table 5.5. Misperception type has a significant effect on economy scores when using Simple Ratio ($F(1, 163) = 13.672, p < 0.003$), Frugality ($F(1, 163) = 10.648, p < 0.002$), and Relative Efficiency ($F(1, 163) = 5.976, p < 0.02$). Exploitation is the only metric not to show any significant effect of misperception type ($F(1, 163) = 1.157, p = 0.211...$).

Inventory size is a significant factor for both Simple Ratio ($F(1, 163) = 389.04, p < 2.2 \times 10^{-16}$) and Exploitation ($F(1, 163) = 124.917, p < 2 \times 10^{-16}$), but it is not significant for Frugality ($F(1, 163) = 82.953, p < 2.92 \times 10^{-16}$) nor for Relative Efficiency ($F(1, 163) = 0.652, p < 0.43$). There is no significant interaction between misperception and inventory size for any of the metrics, except Simple Ratio ($F(1, 163) = 13.672, p < 0.05$).

These results show that when languages are transmitted, via iterated learning, under conditions where sound change influences classes of sounds, the resulting inventories are more economical than they would be if sound change targeted individual segments. This is consistent with Hypothesis 3, which said that feature economy is an emergent consequence of the way sound change operates, as opposed to it being an inherent property of phonological systems.

In simulations with class-level misperceptions, economy scores generally rise because sound changes are affecting classes of sounds, as predicted by Hypothesis 3. For example, suppose there is a simulation with an inventory with a set of three voiceless stops /p, t, k/ in

154
the initial generation. Suppose further that a class-level intervocalic voicing misperception is active. If all three voiceless stops occur between vowels somewhere in the lexicon, then eventually a set of three voiced stops will appear and the inventory will be /p, b, t, d, k, g/. Since these new stops are minimally different from the old ones, differing only by [voice], economy will increase because 3 new sounds are added at the cost of only a single feature. If the feature [voice] is already in use for sounds outside of this stop series, then the increase in economy is even greater, because three new sounds are added “for free”, without the cost of an additional feature.

In a simulation with segment-specific changes, the evolution of the inventory will be different. Assuming the same voiceless stop inventory of /p, t, k/, three new misperceptions will be generated, each of which affects a different feature. It might be that /p/ becomes /f/ between vowels, while /t/ becomes /tʰ/ and /k/ becomes /k’/ in the same environment. If all three segment-specific change occur, then the inventory will not achieve greater economy. Three new sounds will enter the inventory at the cost of three new features.

Toward the end of the simulations, Frugality and Relative Efficiency rise slightly in the segment-specific condition. This rise probably has to do with growth in inventory size. When the inventory grows large enough, the feature space is saturated and even changes targeting randomly-selected features will end up creating sounds that share a feature with an existing sound simply by chance.

Additionally, it is possible for multiple segment-specific changes to have the cumulative effect of class-level changes. For example, a class-level misperception that would have affected /p, t, k/, might turn into segment-specific misperception that changes the [cont] value of /p/ in the first generation, but not the [cont] values of /t/ and /k/. In the second generation, there might be a segment-specific misperception that is generated which changes the [cont] value of /t/ (but not /p/ or /k/), and in the third generation another might be generated changing the [cont] value of /k/ (but not /p/ or /t/). A class-level misperception has effectively occurred over the course of three generations. This is probably a rare event, especially in three consecutive generations, but it is plausible that this happens by chance to at least some sound class over a long number of generations.

Another factor is that the randomly generated segment-specific misperceptions can target any arbitrary feature, creating combinations of features that would never occur in the real world, and giving these changes perhaps an “unfair” advantage over the class-level ones which were more carefully crafted to avoid these unnatural outcomes. For example, it is possible for a segment-specific change to raise the [high] value of a [−high, +low] sound. This could result in a [+high, +low] sound, which is a highly unrealistic sound that could nonetheless increase the economy of an inventory.

Overall, however, these result seem to generally support Hypothesis #3 because (a) economy has been shown to increase due to the way sound change operates and (b) class-level changes lead to inventories with higher economy scores, compared to segment-specific changes.
Chapter 6

Conclusion

In this dissertation, I introduced three hypotheses about how sound change shapes consonant inventories, and tested these hypotheses through computer simulation.

The first hypothesis was that inventory size is related to phonotactic complexity. Languages with more complex phonotactics will tend to develop larger inventories, while languages with more restrictive phonotactics tend to develop smaller inventories. This is because sound change is (mostly) context-sensitive, and phonotactics define the set of possible contexts in a language. Having more possible contexts in a language means that there is a greater diversity of sound changes that could occur. As sounds introduced through change become phonologized, the inventory grows.

This hypothesis was tested by running a large set of simulations grouped into different phonotactic categories. All simulations were initialized with randomly-generated inventories of the same size. The same set of potential misperceptions was used for each simulation. The outcome was that inventories of languages restricted to maximally CV syllables grew the least. Languages with maximally CVC syllables grew into larger inventories, and the largest inventories were found among languages with CCVCC syllables. These results support Hypothesis #1.

The second hypothesis concerned the frequency of consonants across languages. Sounds are not evenly distributed, and some are far more common than others. Small inventories tend to be made up of just the most common sounds, while large inventories contain rare or unique sounds (Lindblom and Maddieson 1988, Maddieson 2011, see Section 4.2).

The existence of cross-linguistically common sounds was hypothesized to be due to the existence of context-free sound changes, which, by definition, apply in inventories of any size. Smaller inventories tend to be made up primarily of the most common sounds, because they have limited phonotactic contexts (by Hypothesis #1). Large inventories have more, and more diverse, phonetic contexts, leading to the evolution of a more diverse array of sounds.

This hypothesis was tested in a way similar to the first one, by running a large number
of simulations. Simulations were initialized with randomly generated sets of segments, and all simulations used the same set of biases and misperceptions. The outcome was that smaller inventories tended to contain mostly those sounds favoured by bias, and inventories diversify as they grow.

The third hypothesis was about feature economy, which is the tendency for inventories to maximize the ratio between the number of segments and the (minimal) number of features required to contrast them (Clements 2003, Mackie and Mielke 2011). Hypothesis #3 states that feature economy emerges in inventories because sound change is defined over classes of sounds, rather than individual sounds. Over time, this produces inventories with sets of sounds differing by only one feature, which is essentially what feature economy measures.

Testing this hypothesis was done by running two kinds of simulations. In one, the probabilistic biases that underlie sound change were defined to take scope over broad classes of sounds, in another they were defined such that they could only affect specific segments. Feature economy was calculated at each generation of these simulations. A linear regression model, with inventory size and misperception type as predictors and economy scores as dependent variable, showed that misperception type had a significant effect on economy scores for all metrics except Exploitation.

These results lend support to the theory that typology is shaped by diachronic forces (e.g. Blevins 2004). However, in contrast to most of the existing research, which tends to focus on specific sound changes, this dissertation has taken a higher-level approach by simulating multiple interacting changes over many generations of language transmission.

The results are also demonstrate how the concept of selection for learnability (Brighton et al. 2005) can be applied to the study of phonological inventories. The sounds that an inventory has are those which are most likely to be successfully retransmitted over time. This gives us a way of understanding certain properties of inventories in a non-teleological framework.

The simulation software designed for this dissertation, PyILM, was built to be open-ended, and could be modified to potentially study other phenomena. There are several changes that could made to the code to increase its utility. For instance, PyILM was constructed with the intention of studying the evolution of consonant inventories, but it could be extended to vowel inventories. Vowel systems were not included in the current study because the way that they change over time seems to be quite different from consonants. In particular, vowel systems show an effect of dispersion (e.g. de Boer 2002), where vowels tend to spread out over the available phonetic space. This is the opposite of the feature economy effect seen in consonant inventories, where a small number of features are re-used. Vowels are also known to undergo chain shifts, which occur much less often in consonant inventories. It would be ideal to update PyILM so that both vowel and consonant evolution can be simulated.

There are also several improvements that could be made to the way that misperceptions are modeled. Currently, only one feature at a time can be affected by misperception, but it would be useful to increase that number. Additionally, it would be convenient to
have features “linked” in some way, such that misperceptions targeting one feature would naturally include another feature (e.g., a misperception that makes a consonant more or less nasal should also make it more or less sonorant).

It would also be useful to diversify the changes that can be modeled as misperceptions. For instance, changes that result in metathesis may be the result of misperception (e.g., Blevins and Garrett (2004)) and could be modeled. Changes could also be non-local, and target segments further away, in order to simulate the evolution of harmony patterns.

Modeling epenthesis and deletion would be very useful, and would have the biggest impact on the results reported in this dissertation because of the potential to disrupt the phonotactics. In particular, the results reported for the tests of Hypothesis #1 and Hypothesis #2 both rely heavily on phonotactics as a main factor, and things may come out differently if phonotactic patterns are not frozen. For example, suppose there is a language with strictly CV syllables, and suppose there is the possibility for vowel reduction/deletion. This means that a CVCV sequence could become a CCV sequence. This puts two consonants adjacent to each other, something that is normally impossible given CV-only phonotactics, and it creates the potential for misperceptions which would otherwise only apply in languages with more complex syllable structures.

Many of the parameters in the simulations are fixed ahead of time, and it would be an improvement if at least some of these values could be more flexibly adjusted over the course of a simulation. In particular, it would be good to have misperception probabilities and salience values be affected by other factors. For instance, functional load plays a role in change, such that sounds which carry a higher functional load are less likely to undergo change (Bouchard-Côté et al. 2013) and avoidance of homophones may be a factor in inhibiting change (Blevins and Wedel 2009). The learning algorithm is another place for improvement. Agents have a few parameters that are set by hand, such as the threshold for deciding if two sounds are distinct or not. Ideally, this is information that agents could learn from data.

Finally, an improvement of a different kind would be to have more of a social environment in the simulations. Currently, PyILM has only one speaker agent and one listening agent per generation. Having a larger population would make it possible for other kinds of sound changes to be modeled. For instance, in a larger population, pronunciations can be considered to be more or less prestigious, and agents can adopt or reject certain pronunciations based on social relationships.

Despite the limitations of PyILM, it still produced interesting and useful results. This, to some extent, makes the results even more interesting. While it might be expected that only a simulation including linguistically significant effects such as notions of contrast, true allophony, or social factors, might be required, PyILM shows that even with very simple assumptions, it is possible to simulate the emergence of phonological patterns in inventories.
Bibliography

Abdel-Massih, E. T.: 1971, Tamazight verb structure: A generative approach, Indiana University, USA.


URL: [http://wals.info/chapter/1](http://wals.info/chapter/1)


URL: [http://wals.info/chapter/6](http://wals.info/chapter/6)


URL: [http://www.theguardian.com/world/2006/feb/12/theobserver.worldnews12](http://www.theguardian.com/world/2006/feb/12/theobserver.worldnews12)


URL: [https://oc-web.library.ubc.ca/cIRcle/collections/24/items/1.0228114](https://oc-web.library.ubc.ca/cIRcle/collections/24/items/1.0228114)


URL: [http://phoible.org/](http://phoible.org/)
URL: http://www.hum.uib.no/a/moren/palauannew.pdf


URL: http://asjp.cldl.org/


