A Python Library for Historical Comparative Linguistics

Steven Moran, Johann-Mattis List

A Python Library for Historical Comparative Linguistics

Steven Moran[1] & Johann-Mattis List[2]

[1] Research Unit *Quantitative Language Comparison*
Ludwig-Maximilians University Munich
steve.moran@lmu.de
[2] Institute for Romance Languages and Philology
Heinrich Heine University Düsseldorf
listm@phil.hhu.de

August 26, 2012

Talk Map

1. Language History
2. Traditional Language Comparison
   - Workflow
   - Cognates
   - Similarity
3. Quantitative Language Comparison
   - Workflow
   - Issues
   - QLC and LingPy
4. Examples
   - Preprocessing
   - Alignments
5. Conclusion

Historical Linguistics

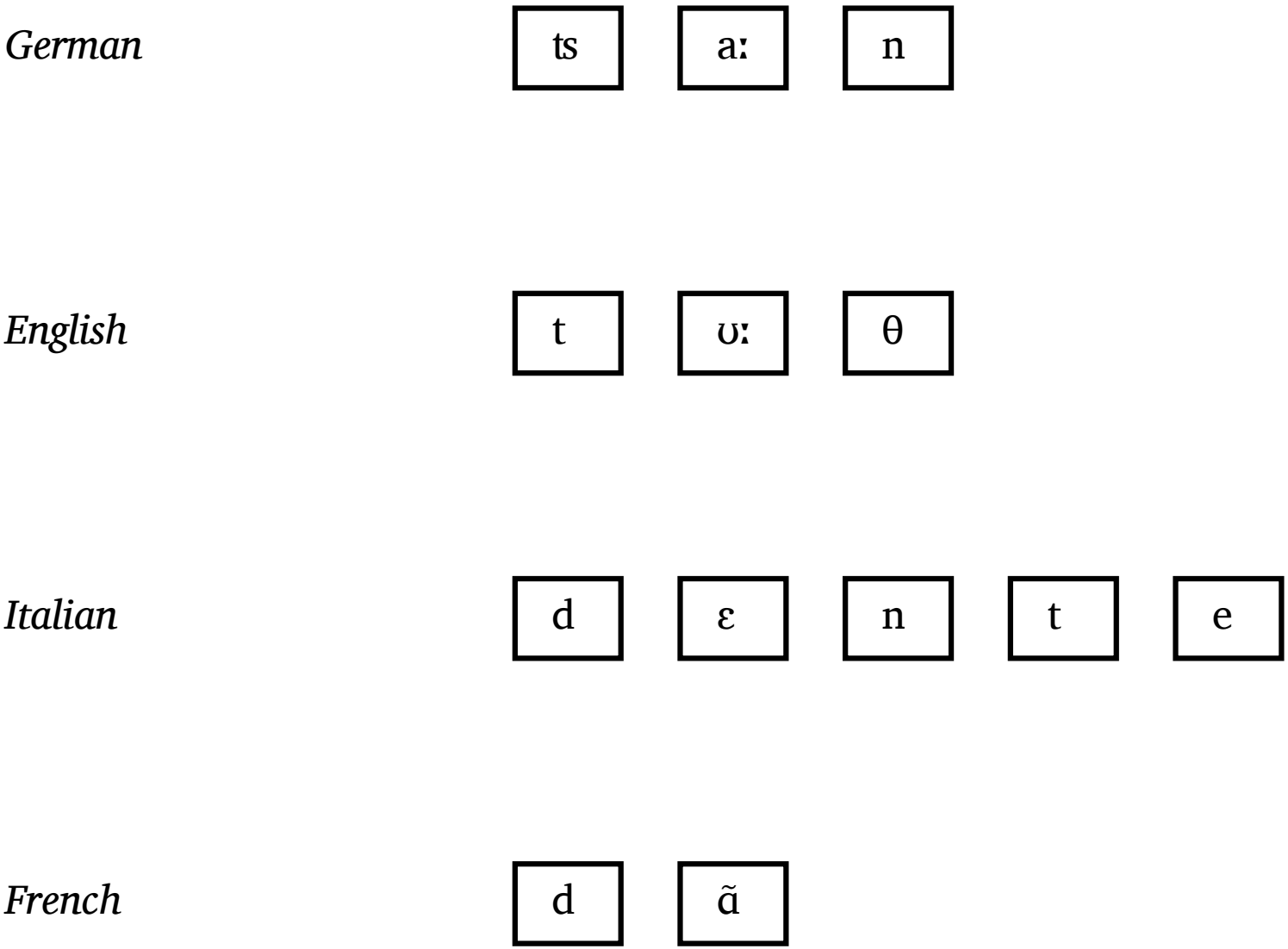
Language History

- Similar to species in biology, languages also evolve. Words are lost, new words are gained, and also the pronunciation of all words changes slightly from day to day.
- During its history, a language may split into two or more descendant languages when the speakers separate and their languages keep on changing independently.
- To uncover, how the languages changed into their current shape is one of the major tasks of historical linguistics.

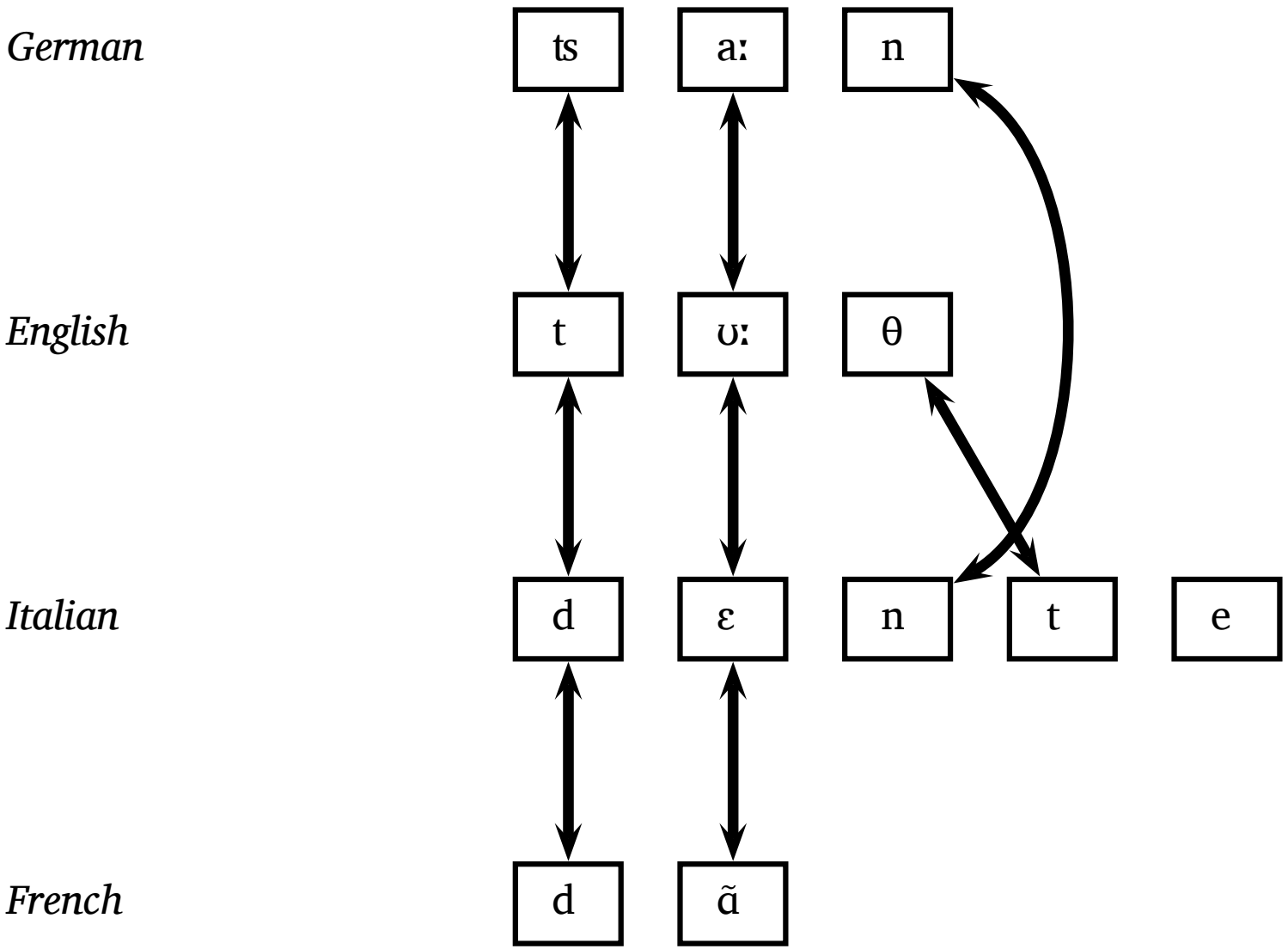
Uncovering Language History

- There are only a few languages whose history is directly reflected in written sources.
- For the majority of the 6909 languages spoken today (Lewis 2009), we would not know anything about their past if we didn't have methods to infer their history.
- In order to uncover language history, the languages spoken today are manually searched for traces of common origin.
- Finding these traces, however, is an extremely complicated task: it took scholars more than 50 years to prove that Armenian is an Indo-European language...

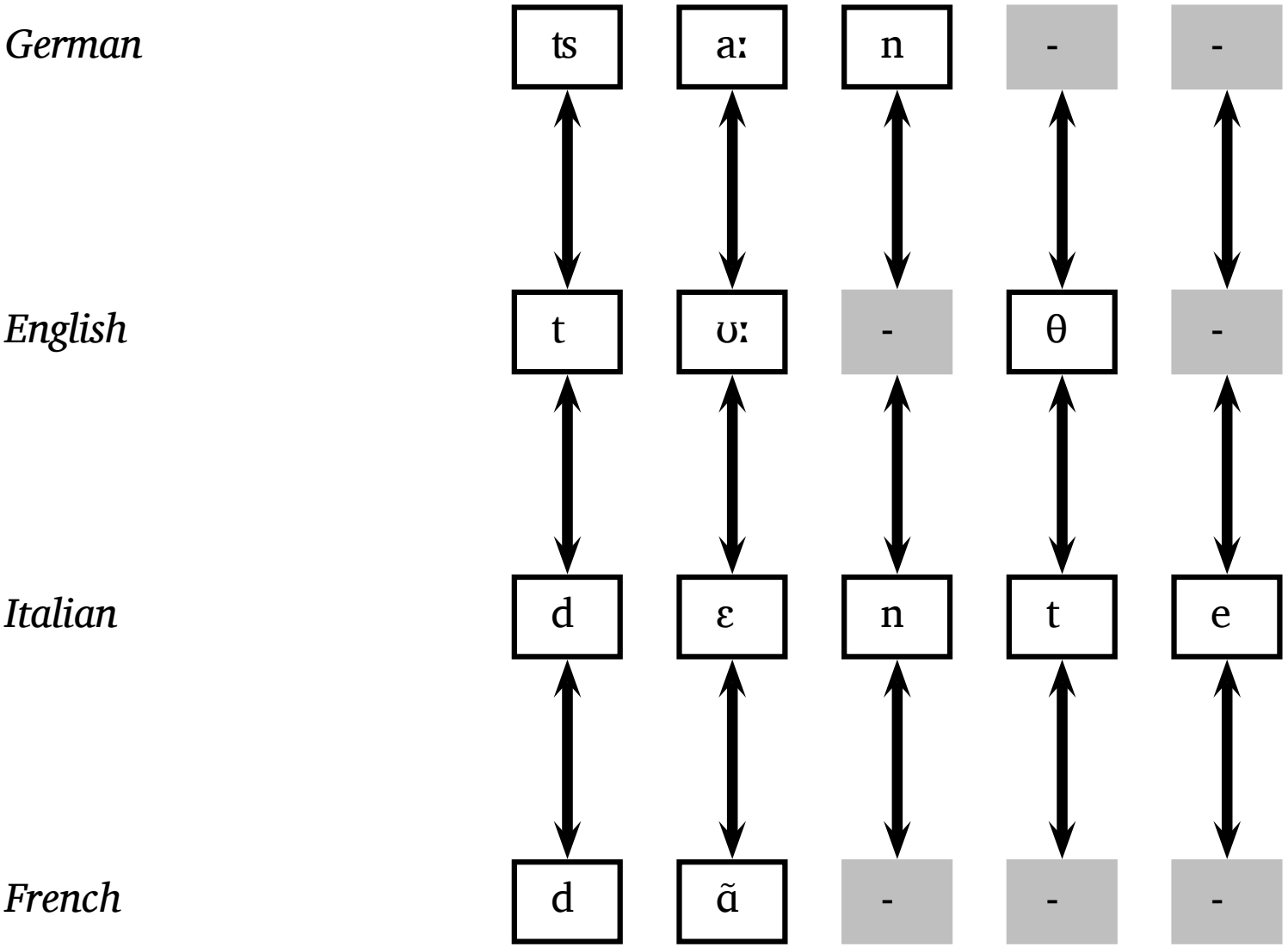
Constructing Historical Scenarios



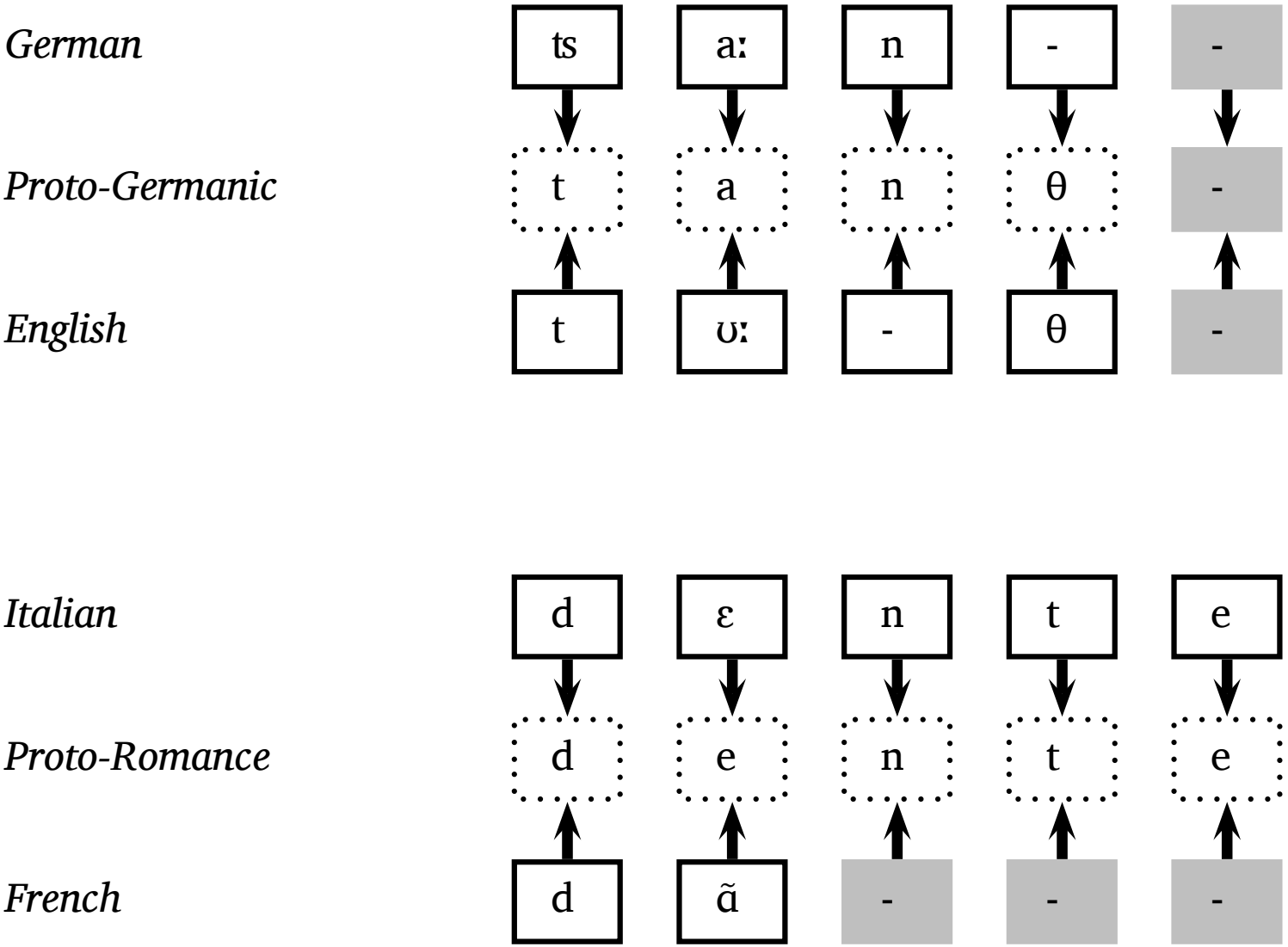
Constructing Historical Scenarios



Constructing Historical Scenarios



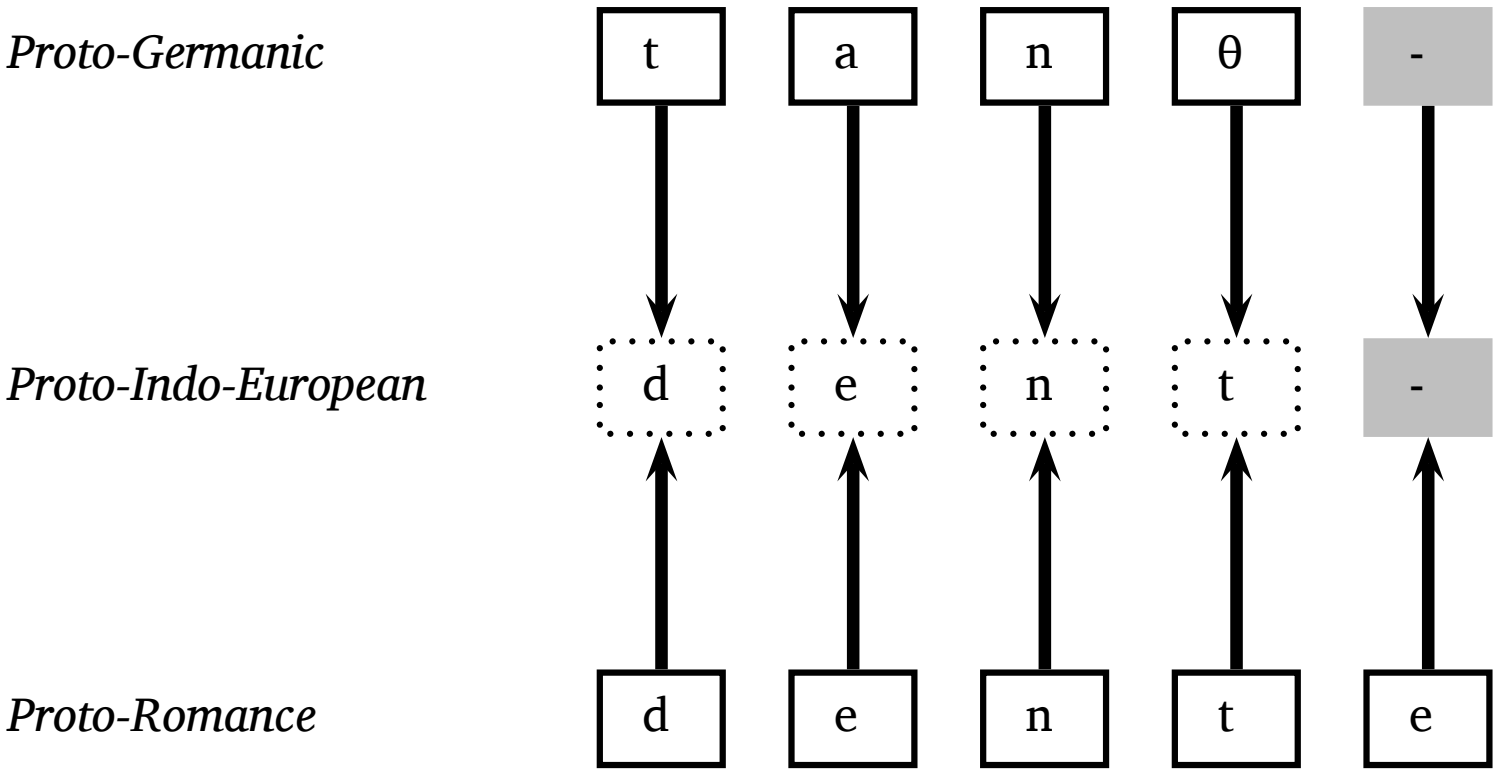
Constructing Historical Scenarios



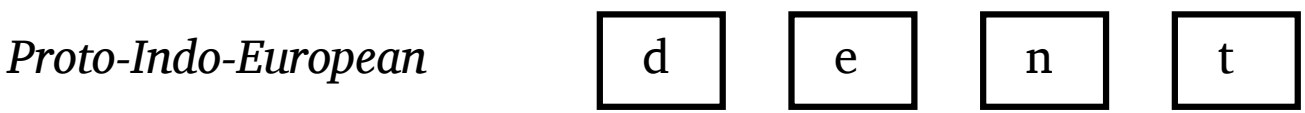
Constructing Historical Scenarios

Proto-Germanic	t	a	n	θ	-
Proto-Romance	d	e	n	t	e

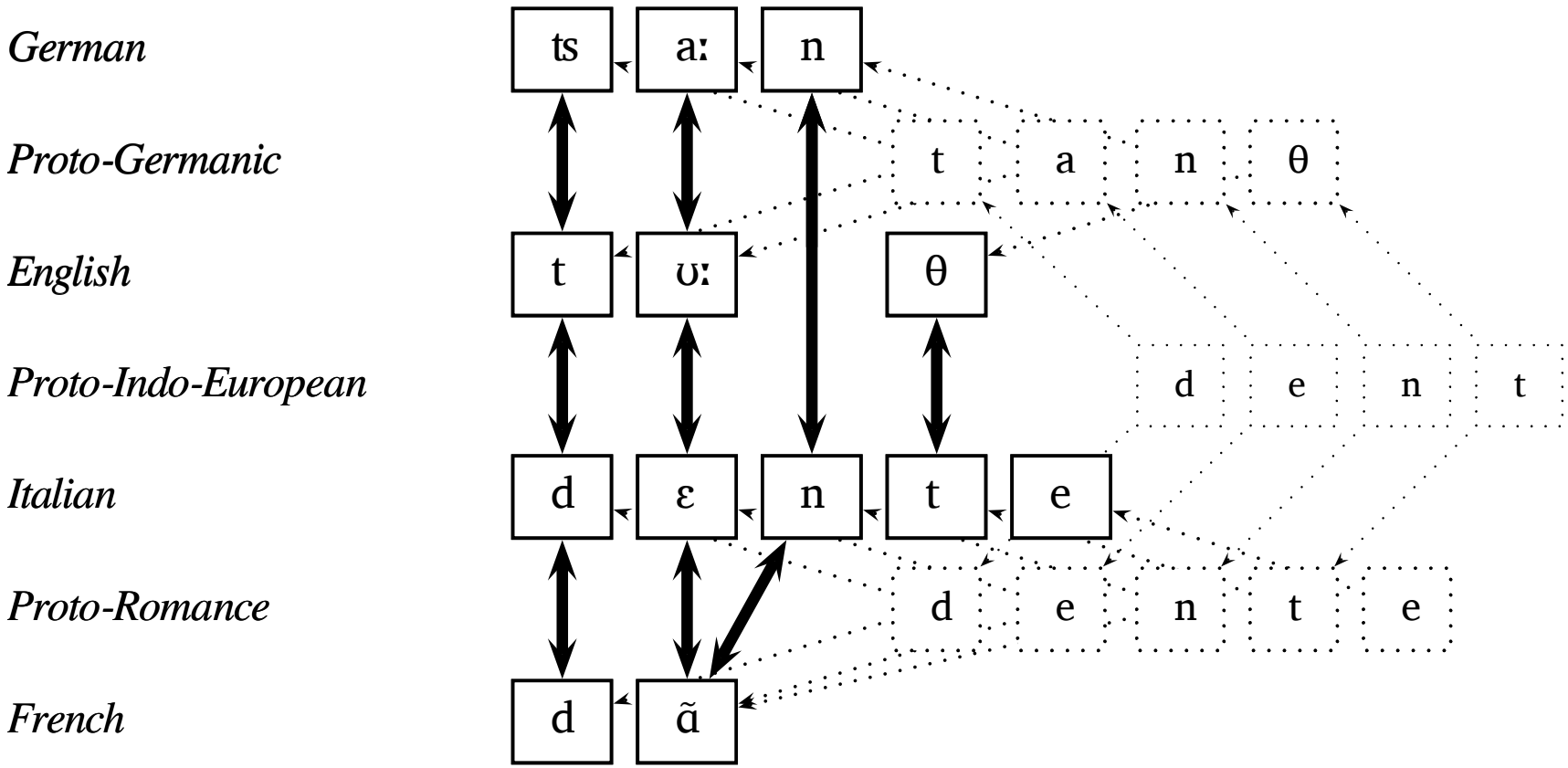
Constructing Historical Scenarios

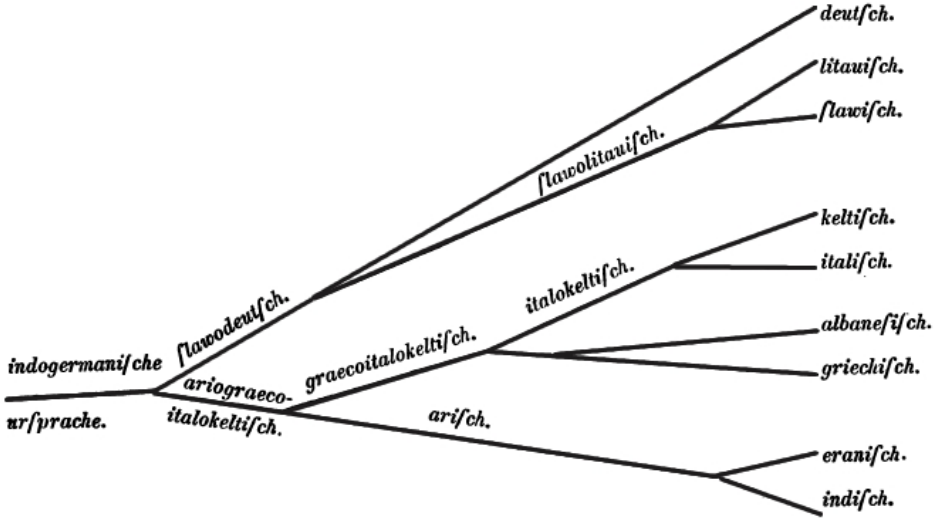


Constructing Historical Scenarios

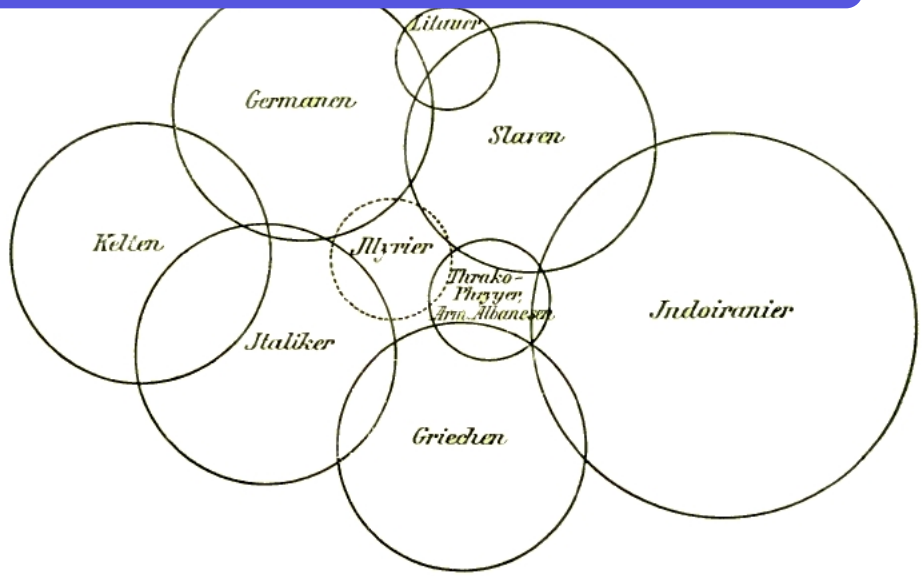


Constructing Historical Scenarios

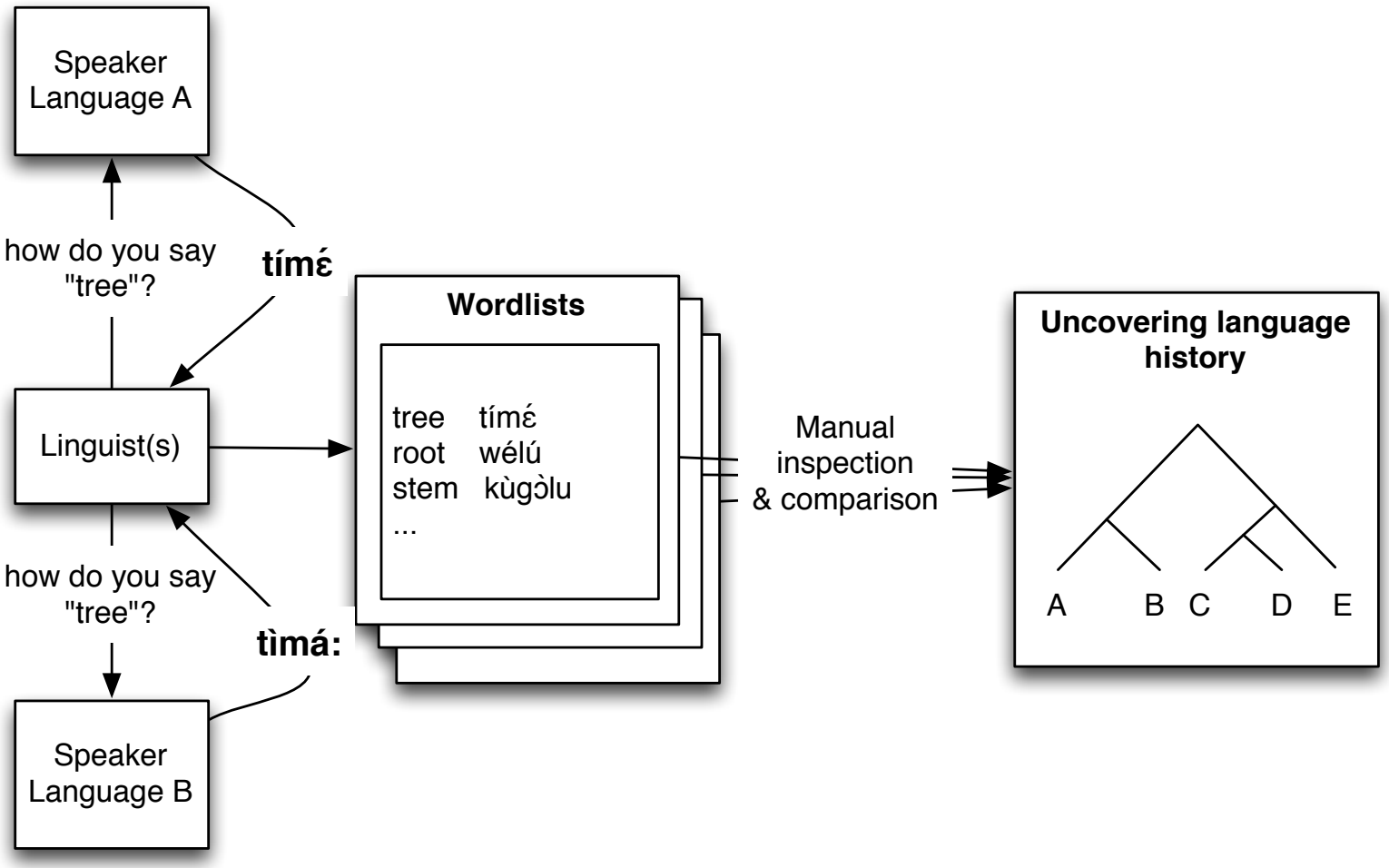




Traditional Language Comparison



Slide 1 – Workflow



Cognate Detection

- The most crucial part of language comparison is the detection of cognates.
- Cognates are words from different languages that go back to a common ancestor word (compare German *Hand* and English *hand*).
- Cognate words exhibit a specific kind of similarity which does not necessarily show up in form of surface resemblances of the sounds the words are made of, but rather in structural similarities of cognate words.
- This kind of similarity is not easy to detect: German *Zahn* and English *tooth*, for example, are cognate, while Greek *mati* and Malay *mata* are not...

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

$$j = l$$

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

Meaning	Italian	French
“tear”	lakrima	laʀm
“tongue”	liŋgwa	lɑ̃g
“moon”	luna	lyn

$j = l$

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

j = l

Meaning	Italian	French
“tear”	lakrima	laʀm
“tongue”	liŋgwa	lɑ̃g
“moon”	luna	lyn

l = l

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

Meaning	Italian	French
“tear”	lakrima	laʀm
“tongue”	liŋgwa	lɑ̃g
“moon”	luna	lyn

j = l

l = l

pj = pl , l = l

Sound Correspondences

Meaning	Italian	French
“square”	pjatsːa	plas
“feather”	pjuma	plym
“flat”	pjano	plã

Meaning	Italian	French
“tear”	lakrima	laʀm
“tongue”	liŋgwa	lɑ̃g
“moon”	luna	lyn

j = l

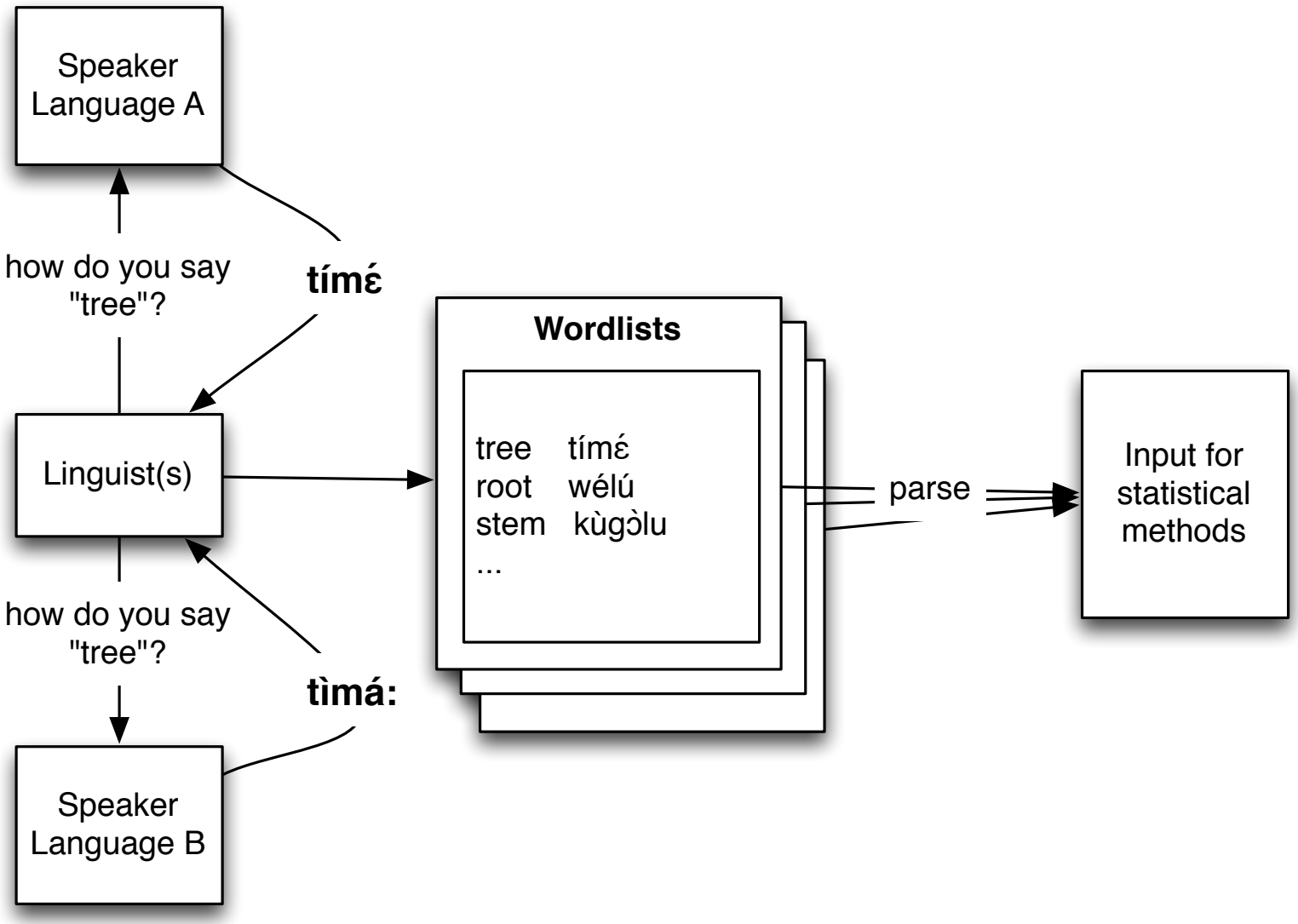
l = l

pj = pl , l = l

Detecting regular sound correspondences can be compared to the task of finding a cipher that relates a source to a cipher text. The only problem is: **It is much harder than that, since source and cipher text are badly maintained...**

Quantitative Language Comparison

Workflow


Speaker
Language A
how do you say
"tree"?
tímɛ́
Linguist(s)
how do you say
"tree"?
tìmá:
Speaker
Language B
Wordlists
tree tímɛ́
root wélú
stem kùgɔ̀lu
...
parse
Input for
statistical
methods

Biological Parallels

In both biology and historical linguistics ...

- sequences, i.e. ordered collections drawn from a fixed set of characters, constitute the basic unit of replication.
- sequence comparison is of great importance.
- phylogenetic trees are a basic classification scheme.

Biological Parallels – Caveats

Problems in historical linguistics and biology are similar, but ...

- the amino acid alphabet for proteins has only 20 characters, while there are more than 2000 different sounds in the languages of the world...
- biological sequences are very long while linguistic ones are very short...
- in biology, the alphabet remains stable during evolution, while it changes constantly in language history...

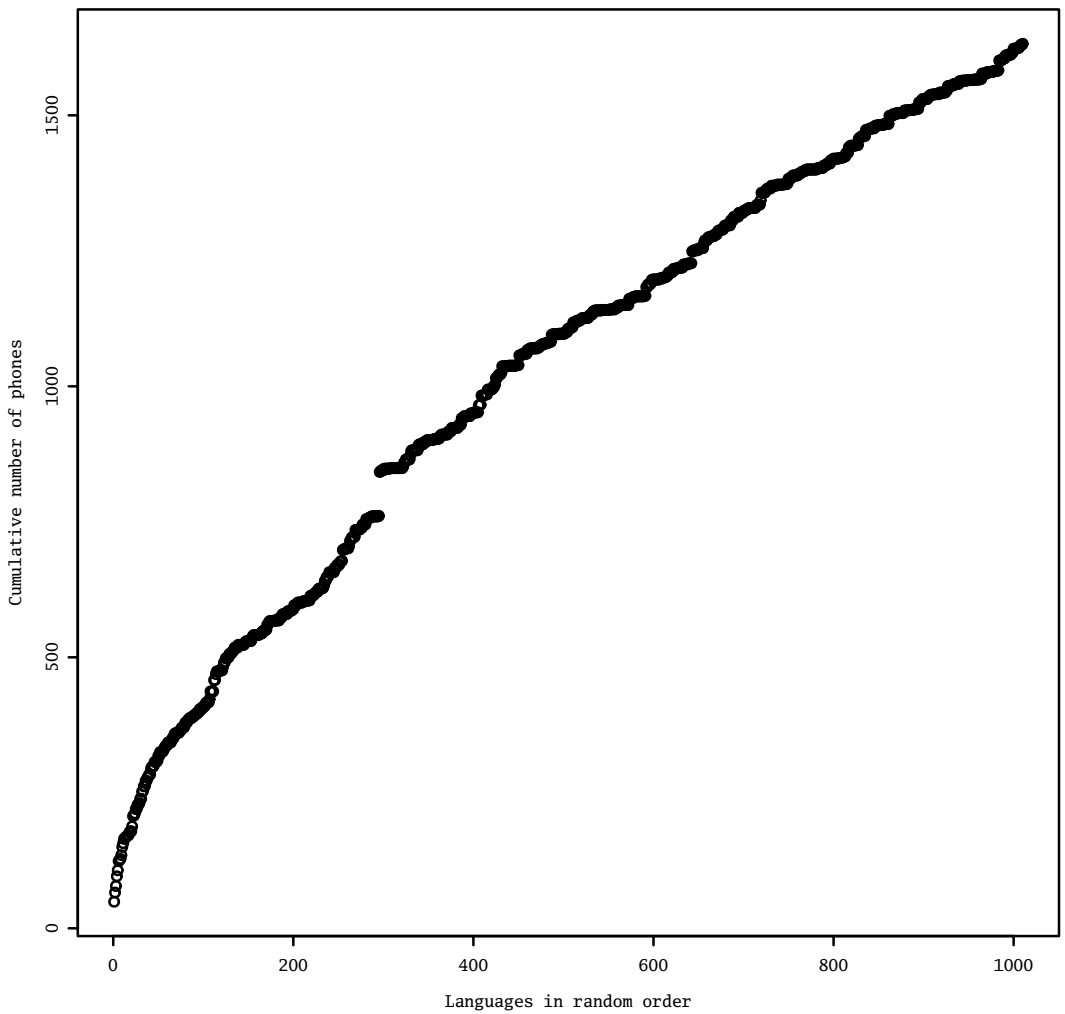
Biological Parallels – Caveats

Problems in historical linguistics and biology are similar, but ...

- the amino acid alphabet for proteins has only 20 characters, while there are more than 2000 different sounds in the languages of the world...
- biological sequences are very long while linguistic ones are very short...
- in biology, the alphabet remains stable during evolution, while it changes constantly in language history...

Biological algorithms were designed for long sequences drawn from small alphabets. In quantitative language comparison we need algorithms for short sequences drawn from large alphabets. While biologists can model their sequences in ASCII characters, linguists cannot do without Unicode!

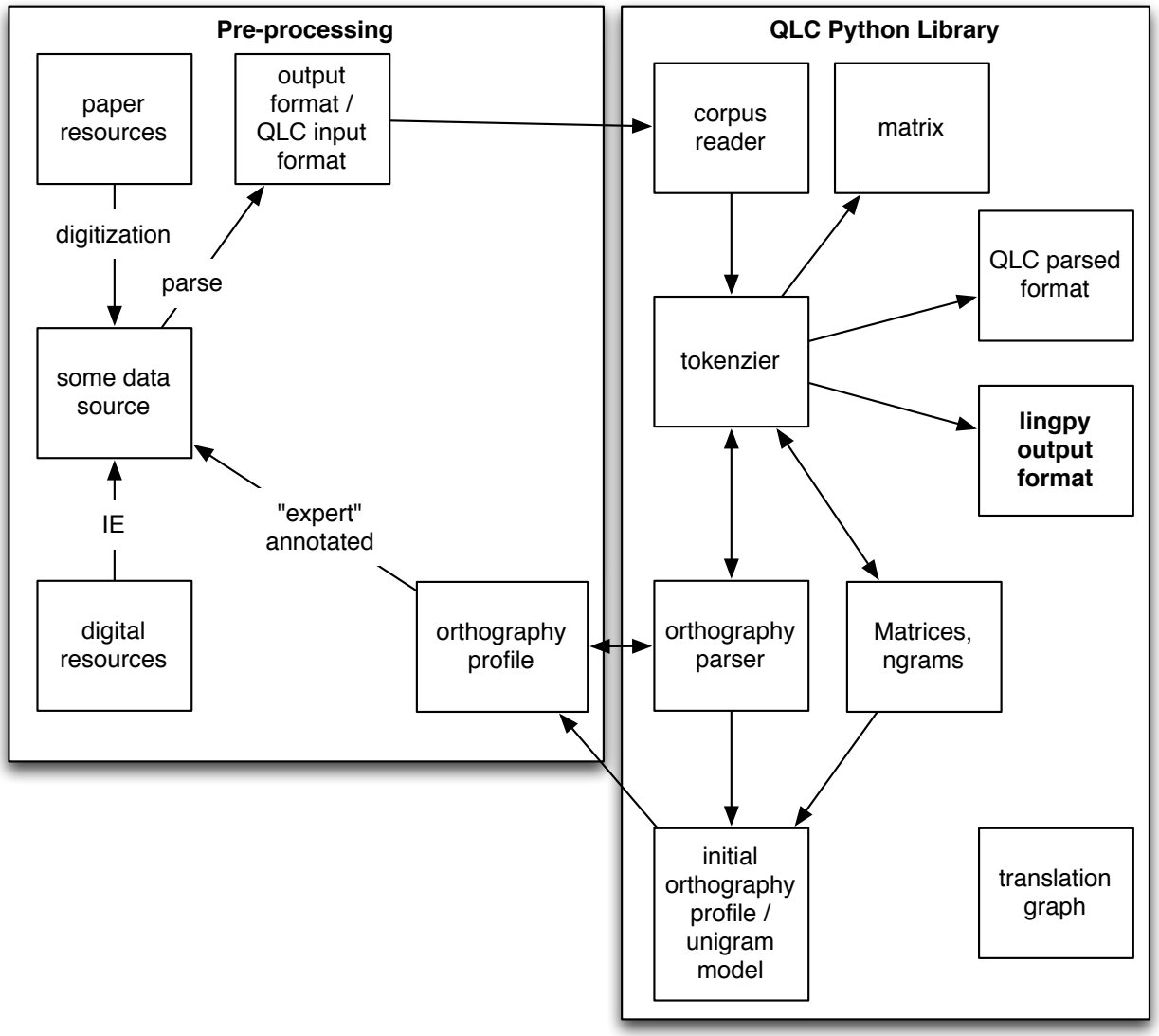
Cumulative number of segment types vs languages in random order



QLC

- QLC is a Python library for undertaking quantitative language comparison.
- QLC focuses on parsing linguistic data, orthographic tokenization and quantitative methods for language comparison.
- QLC uses the Python libraries “regex” (Matthew Barnett), SciPy, NumPy and LingPy.
- `https://github.com/pbouda/qlc`

QLC


Pre-processing
paper resources
output format / QLC input format
digitization
parse
some data source
IE
"expert" annotated
digital resources
orthography profile
QLC Python Library
corpus reader
matrix
QLC parsed format
tokenzier
lingpy output format
orthography parser
Matrices, ngrams
initial orthography profile / unigram model
translation graph

LingPy

- LingPy is a Python library for automatic tasks in historical linguistics.
- LingPy mainly concentrates on statistical analyses of languages, providing methods for
  - sequence modelling,
  - pairwise and multiple sequence alignment,
  - automatic cognate detection, and
  - plotting routines.
- LingPy makes use of NumPy and NetworkX. Time-consuming tasks are implemented in C++ (integrated using Boos) and Cython.
- `http://lingulist.de/lingpy`

v o l - d e m o r t

v - l a d i m i r -

v a l - d e m a r -

Examples

Orthography profile

- A string must be tokenized into Unicode code points
- Tokenization is required because sequences of code points can differ in their visual and logical orders
- Unicode normalization reorders code points into a canonical order
- Combining characters and space modifying letters are joined
- An orthography profile specifies graphemic sequences, e.g. <s> & <h> vs <sh> in <mis.hap> & <mish.mash>

Computational challenges

- Adherence to Unicode IPA:
  - g/g, !/! ...
  - a/ɑ
  - p/p
- Visual versus logical Unicode character ordering (homoglyphs)

ã̲	ã̲
U+0061 + U+0330 + U+0303	U+0061 + U+0303 + U+0330
latin small letter a + combining tilde below + combining tilde	latin small letter a + combining tilde + combining tilde below

Computational orthographic levels

- original string: tsʰõ̰shi

code points (10)	t	s	ʰ	o	̰	̃	́	s	h	i
characters (6)	t	sʰ		õ̰				s	h	i
graphemes (4)	tsʰ			õ̰				sh		i

Computational orthographic levels

1. original string : tsʰõ̀shi
2. code points (10): t s ʰ o ̃ ̰ ̀ s h i
3. characters (6): t sʰ õ̀ s h i
4. graphemes (4): tsʰ õ̀ sh i

Orthography profile example: Leach 1969

#		leach1969 orthography profile (interpretation by QLC)
-,	,	
a,	a,	
á,	a,	
an,	ã,	
án,	ã,	
e,	ɛ,	according to Aschmann extremely rare
é,	ɛ,	according to Aschmann extremely rare
en,	ɛ̃,	according to Aschmann does not exist
én,	ɛ̃,	according to Aschmann does not exist
c,	k,	
ch,	ʧ,	
d,	d,	only in Spanish loan words
..j,	h,	typo

Orthography profile example: Leach 1969

s,	s,	
sh,	ʃ,	according to Aschmann it never occurs word initially
t,	t,	
ts,	ts,	
ty,	tʲ,	
y,	ʒ,	not clear which sound it is, it never occurs word initially
z,	z,	only in Spanish loan words

rules
([a|á|e|é|i|í|o|ó|u|ú])(n)(\s)([a|á|e|é|i|í|o|ó|u|ú]), \1 \2 \4

Sound Classes

Sound Classes

Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).

Sound Classes

Sound Classes

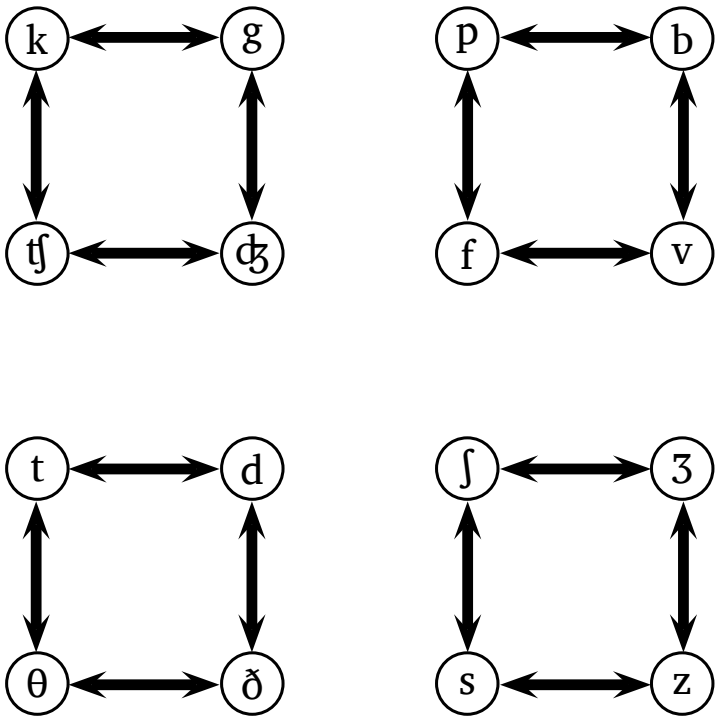
Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).

k	g	p	b
tʃ	dʒ	f	v
t	d	ʃ	ʒ
θ	ð	s	z

Sound Classes

Sound Classes

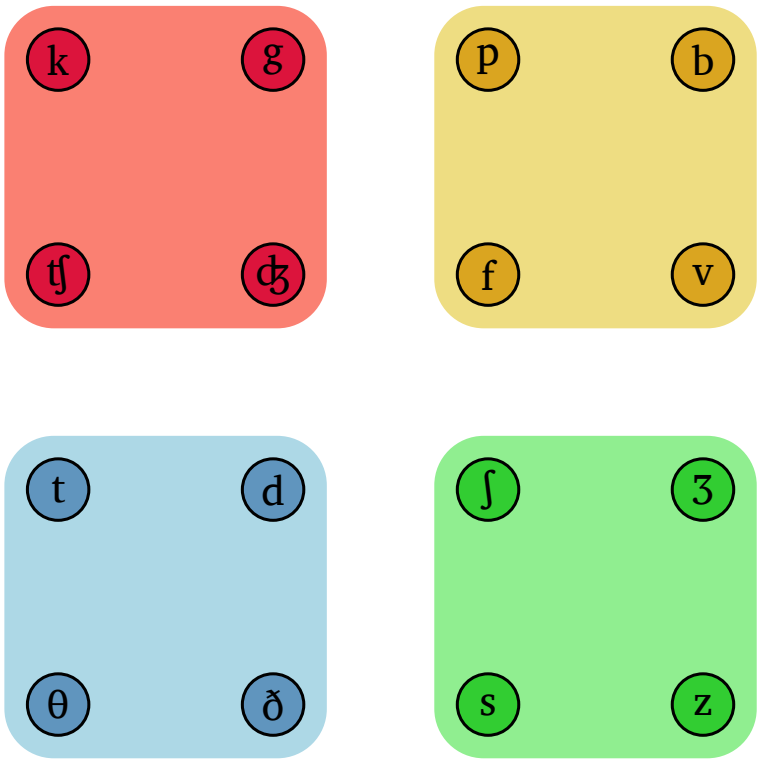
Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).



Sound Classes

Sound Classes

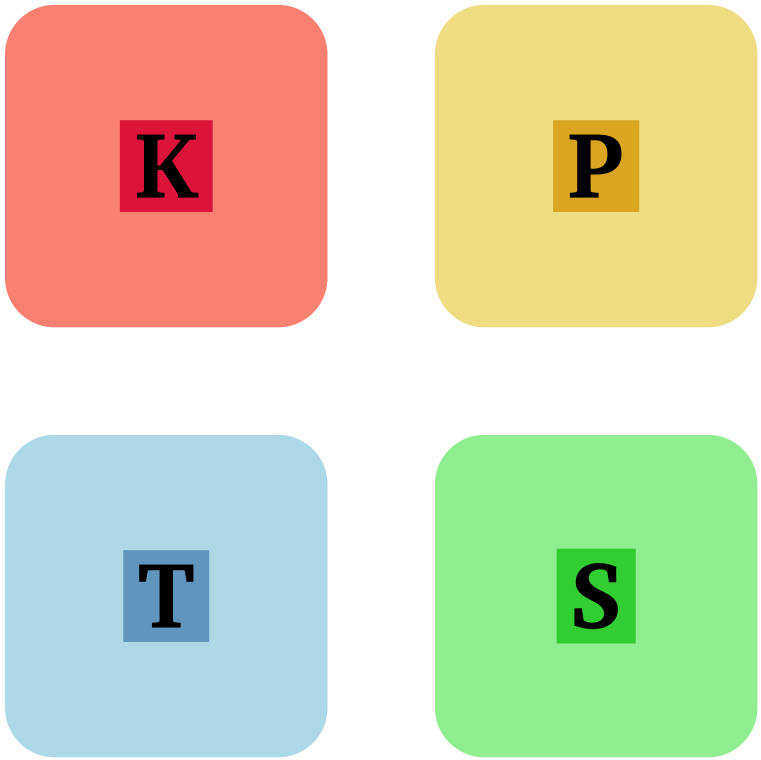
Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).



Sound Classes

Sound Classes

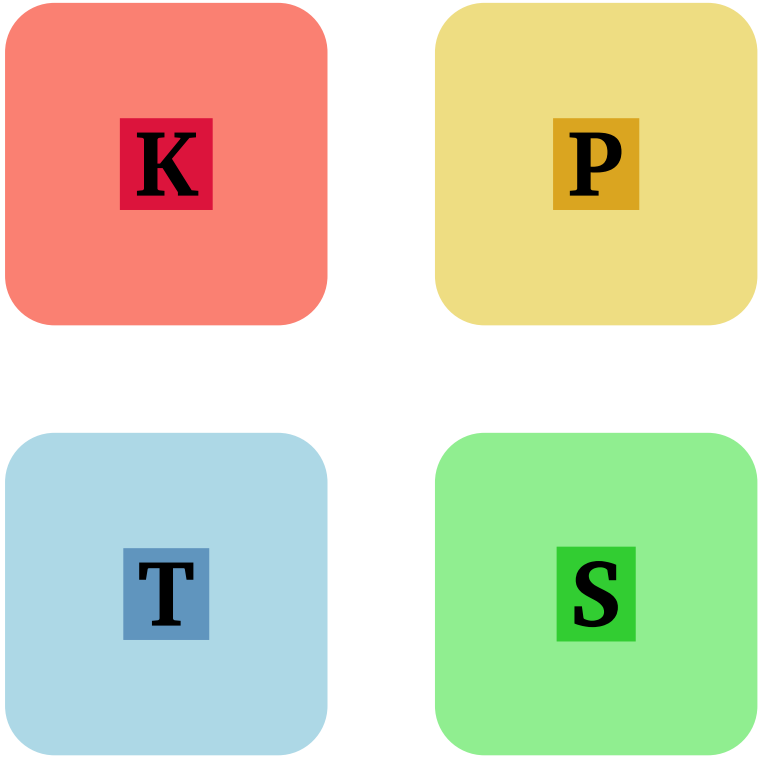
Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).



Sound Classes

Sound Classes

Sounds which often occur in correspondence relations in genetically related languages can be clustered into classes (types). It is assumed “that phonetic correspondences inside a ‘type’ are more regular than those between different ‘types’” (Dolgopolsky 1986: 35).



With the help of sound classes the large number of speech sounds can be reduced to a manageable size comparable to the number of characters used in biological algorithms.

Alignment Analyses

Definition 1

An *alignment* of *n* sequences is an *n*-row matrix in which all sequences are aranged in such a way that all matching and mismatching segments occur in the same column, while empty cells, resulting from empty matches, are filled with gap symbols. (cf. Kruskal 1983)

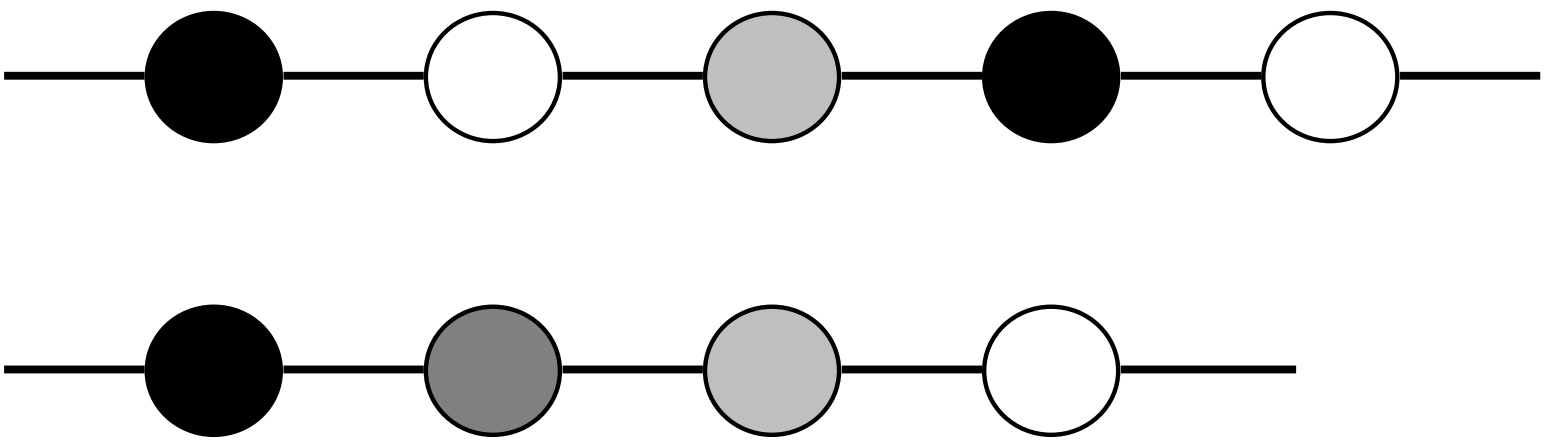
Alignment Analyses

Definition 1

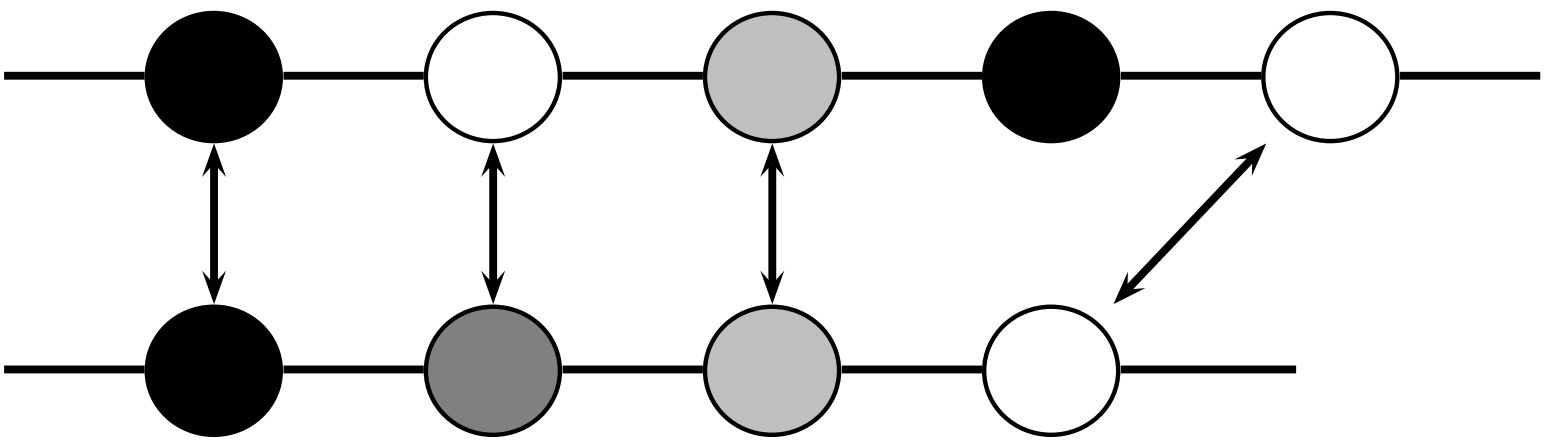
An *alignment* of *n* sequences is an *n*-row matrix in which all sequences are aranged in such a way that all matching and mismatching segments occur in the same column, while empty cells, resulting from empty matches, are filled with gap symbols. (cf. Kruskal 1983)

For reasons of computational complexity, the alignment problem is often split into a pairwise and a multiple alignment problem. Multiple sequence alignments are usually computed on the basis of previously computed pairwise alignments.

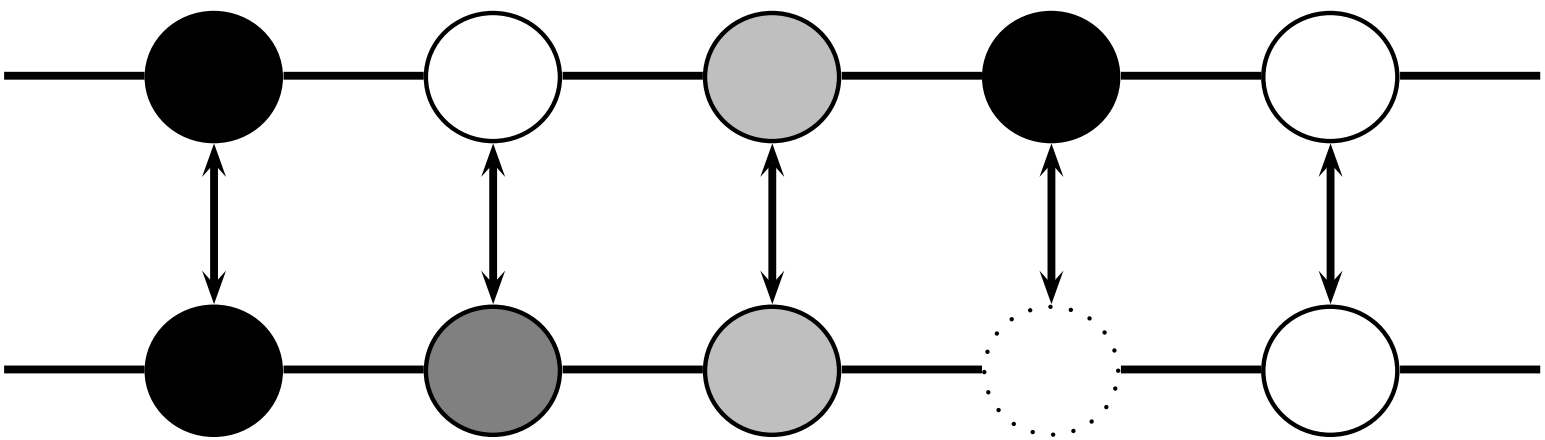
Alignment Analyses



Alignment Analyses



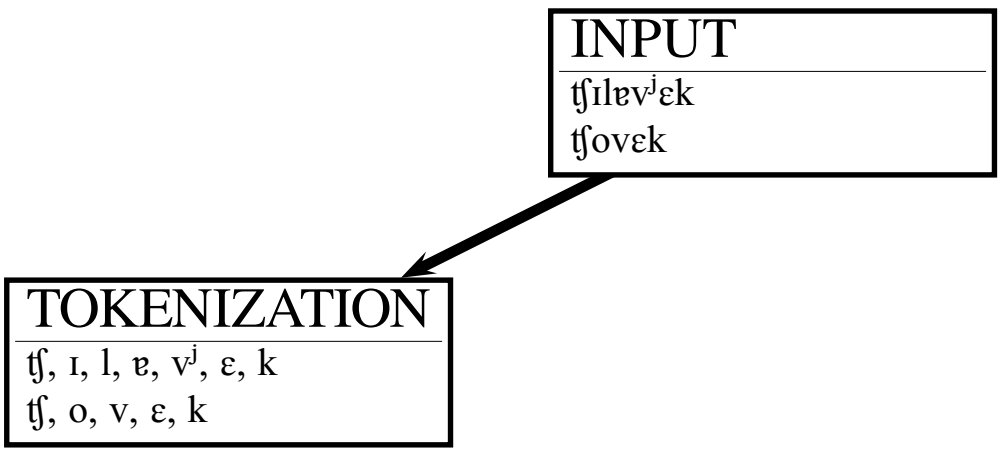
Alignment Analyses



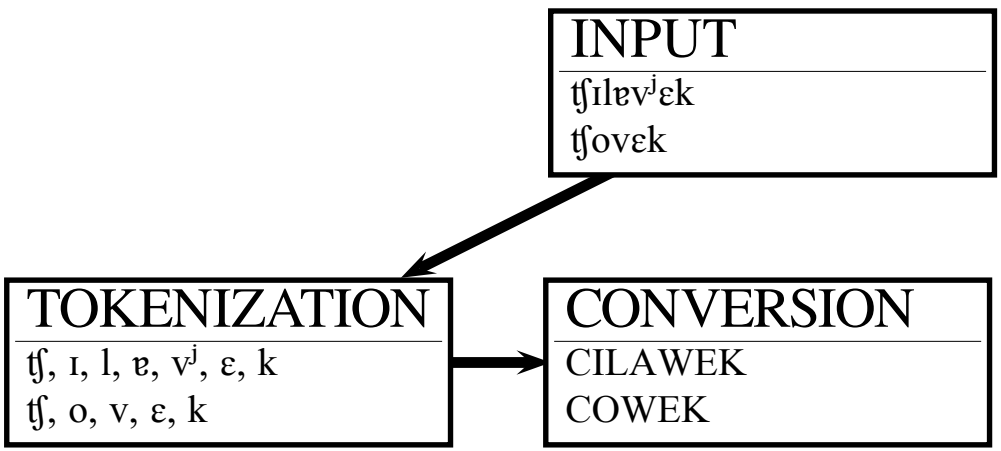
Phonetic Alignment

INPUT
ʧɪlɐvʲɛk
ʧovɛk

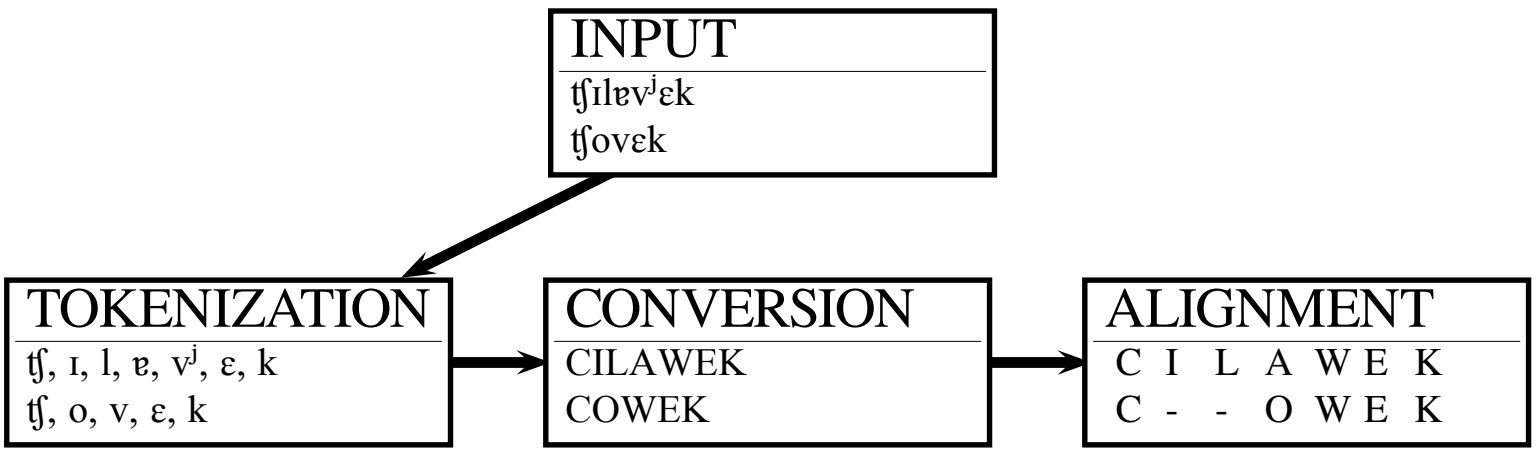
Phonetic Alignment



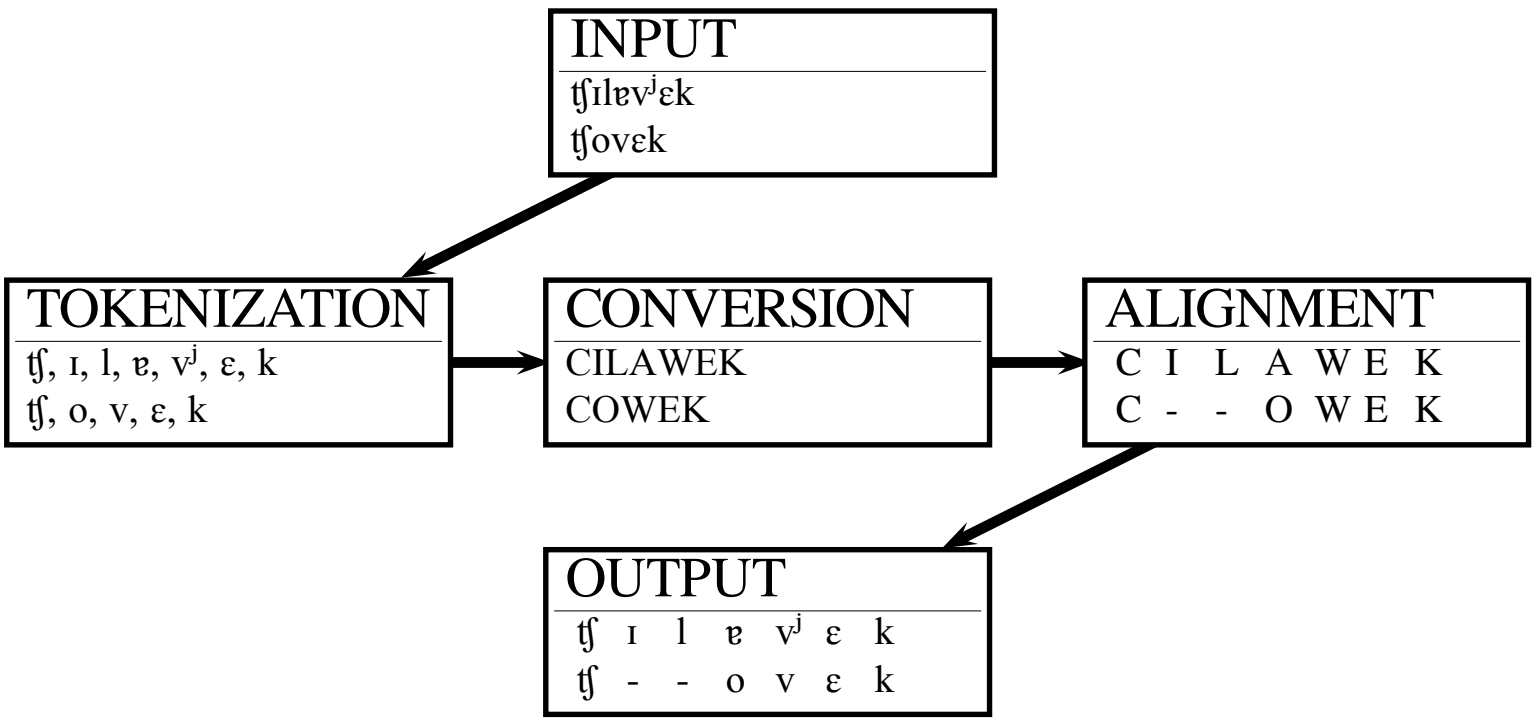
Phonetic Alignment



Phonetic Alignment



Phonetic Alignment



Example

```
>>> from lingpy import *
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kʊ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kʊ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
>>> test.lib_align()
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kʊ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
>>> test.lib_align()
>>> print test
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kɤ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
>>> test.lib_align()
>>> print test
k - u ²¹ - - -
k - ɤ ²¹ - - -
k - u ²¹ - - -
k - u ²¹ - - -
g - u ¹² - - -
k w o ²¹ - - -
q - ẽ ⁵⁵ k ɤ ³³
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kɤ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
>>> test.lib_align()
>>> print test
k - u ²¹ - - -
k - ɤ ²¹ - - -
k - u ²¹ - - -
k - u ²¹ - - -
g - u ¹² - - -
k w o ²¹ - - -
q - ẽ ⁵⁵ k ɤ ³³
>>> print ','.join(test.classes)
KY3,KE3,KY3,KY3,KY2,KY3,KY3
```

Example

```
>>> from lingpy import *
>>> test = Multiple('evobench_91')
>>> print ','.join(test.seqs)
ku²¹,kɤ²¹,ku²¹,ku²¹,gu¹²,ku²¹,ku²¹
>>> test.lib_align()
>>> print test
k - u ²¹ - - -
k - ɤ ²¹ - - -
k - u ²¹ - - -
k - u ²¹ - - -
g - u ¹² - - -
k w o ²¹ - - -
q - ẽ ⁵⁵ k ɤ ³³
>>> print ','.join(test.classes)
KY3,KE3,KY3,KY3,KY2,KY3,KY3
```

Why not use already existing tools?

Well, because

- for a majority of the sequence comparison algorithms used in LingPy there is no proper Python implementation available, thus
  - PyCogent (Knight et al. 2007) and BioPython (Cock et al. 2009) offer only implementations of the most basic algorithms (Needleman-Wunsch, Smith-Waterman) for pairwise alignment,
  - there are no Python implementations of algorithms for multiple sequence alignment that we would know of, and
  - neither the basic progressive strategies (Thompson et al. 1994) for multiple sequence alignment, nor the recent consistency-based algorithms (Notredame et al. 2000) that are implemented in LingPy have made their way into any of these packages.
- most algorithms that have been developed for bioinformatics cannot be directly used to compare linguistic sequences, but have to be adapted to conform to the specific needs of historical linguistics (see List 2012 and the LingPy documentation for details).

Open ended language comparison problems

- How to distinguish between words that are similar because they are historically related and words that are similar by chance?
  - Greek *théos'* [θɛɔs] "god" and Spanish *dios* [diɔs] "god" are similar, yet they are not historically related, going back to different ancestor words.
- How to distinguish between words that have a common origin and words that have been borrowed?
  - English has borrowed more than 40% of its lexicon, mostly from Romance languages.
- How can we standardize the way we represent languages in our programs, how can we effectively pool our strengths in uncovering the unknown history of the languages of the world?
  - Standardization plays an important role in biology. In linguistics, it has been ignored so far.

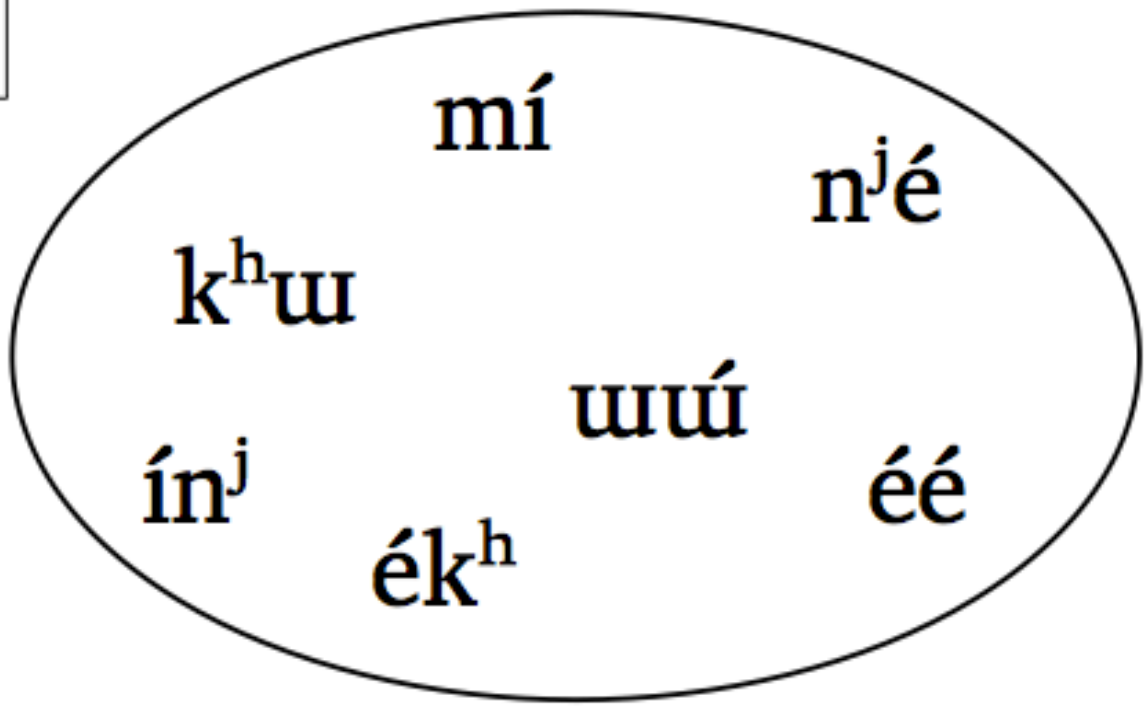
Many thanks to...

The organizers, participants and sponsors of EuroSciPy 2012!
QLCers: Michael Cysouw, Jelena Prokić and Peter Bouda.
And the European Research Council.

“Bag of symbols” approach

- Alignment of sounds in lexical items
- Ignore linear structure of words (mostly)
- Use parallel wordlist to estimate co-occurrences of n-grams
- N-grams that have a high probability of co-occurrence in parallel meaning are interested for historical linguistics

mínʲéékʰɯɯ́



(2-grams)

	Bora	**Muinane**
down	tʃɨnʲe, paári	báari, gíino
bee	íímɯ́ʔóexpʰi, téʔtsʰipa	nɨ̂ɨ̂biɨri, mɨ̂ɨ̂biɨriʔi
sharp	tsʰɯ́ʔxɨβáne	sɨ̂ɨ̂xéβano
…	…	…

Bora	Muinane	Bora	Muinane	Bora	Muinane
#k	#k^{h}	#i	#i	#n	#n
kɨ	k^{h}ɯ	#a	#a	#m	#m
se	ts^{h}ɨ	di	ti	mɨ	mɯ
xe	xɨ	du	to	nɨ	nɯ
ga	k^{w}a	#d	#t	us	ts^{h}ɨ
ba	pa	#s	#ts^{h}	#t	#t^{h}
#b	#p	gi	ʧi	ɨg	ɯk^{w}
e#	ɨ#	ni	ni	#ɸ	#p^{h}

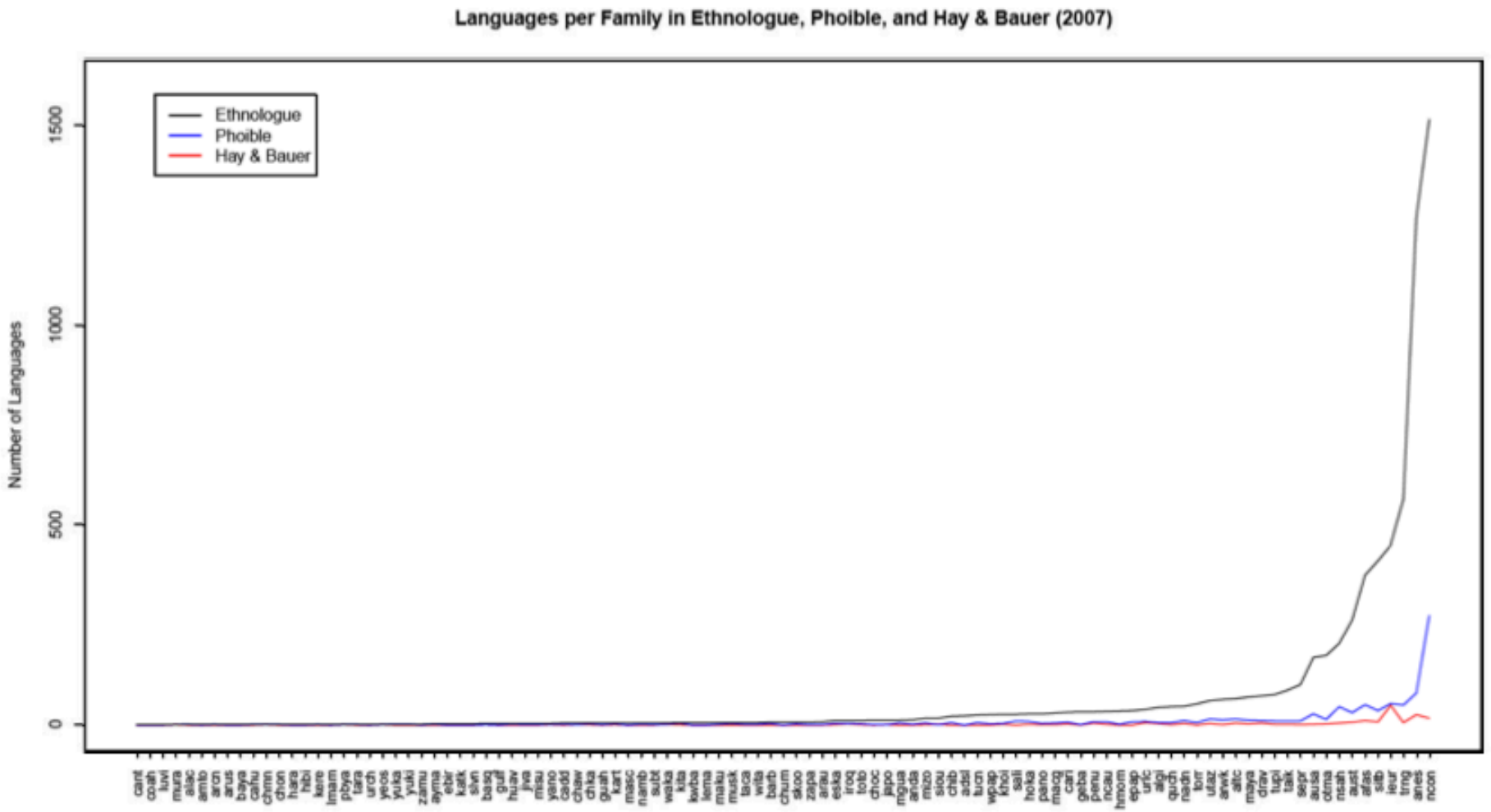
Bigram matching

- Bora “two”: mínjéékhɯɯ
- Muinane “two”: míínokɬ

	#m	mi	ii	in	no	ok	kɨ	ɨ#
#m	22	3	2	2	2	2	2	2
mi	4	12	2	2	5	1	1	1
in^{j}	2	1	5	9	3	1	1	2
n^{j}e	1	1	5	5	4	1	1	2
ee	3	3	3	3	6	2	2	2
ek^{h}	1	2	1	1	4	2	3	2
k^{h}ɯ	2	2	2	2	2	1	23	2
ɯɯ	2	2	3	3	2	2	4	4
ɯ#	2	2	3	2	3	1	3	4

	#m	mi	ii	in	no	ok	kɨ	ɨ#
#m	22	3	2	2	2	2	2	2
mi	4	12	2	2	5	1	1	1
in^{j}	2	1	5	9	3	1	1	2
n^{j}e	1	1	5	5	4	1	1	2
ee	3	3	3	3	6	2	2	2
ek^{h}	1	2	1	1	4	2	3	2
k^{h}ɯ	2	2	2	2	2	1	23	2
ɯɯ	2	2	3	3	2	2	4	4
ɯ#	2	2	3	2	3	1	3	4

Languages per language family



Distribution of languages by speaker population

	Languages		Speakers	
Population range	Count	Percent	Count	Percent
100,000,000 to 999,999,999	8	0.1	2.30 B	38.7
10,000,000 to 99,999,999	77	1.1	2.34 B	39.3
1,000,000 to 9,999,999	304	4.4	952 M	15.9
100,000 to 999,999	895	13.0	283 M	4.7
10,000 to 99,999	1,824	26.4	61 M	1.01
1,000 to 9,999	2,014	29.2	7.7 M	0.13
100 to 999	1,038	15.0	461 K	0.007
10 to 99	339	4.9	12.5 K	0.0002
1 to 9	133	1.9	521	0.00001
Unknown	277	4.0		
Totals	6,909		6 B	

Source: Ethnologue 16 (Lewis 2009)

Major languages geographically