Computational Phylogenetics 2

Quantitative Methods in Historical Linguistics

Dr. Henri Kauhanen / University of Konstanz

23 May 2018

From last time: Exercise

- Here's a feature matrix for four imaginary languages A, B, C and D and four arbitrary binary features F1–F4:

	F1	F2	F3	F4
A	1	1	1	1
B	1	1	1	0
C	1	0	0	0
D	0	0	0	0

1. Form the distance matrix
2. Use UPGMA to figure out the topological tree
3. Use the ultrametricity property to figure out distances on the edges of the corresponding metric tree

Part I

Problems of feature-driven, distance-based methods

Problems?

- Last time, we used a distance-based method (UPGMA) to infer a tree from the following typological data:

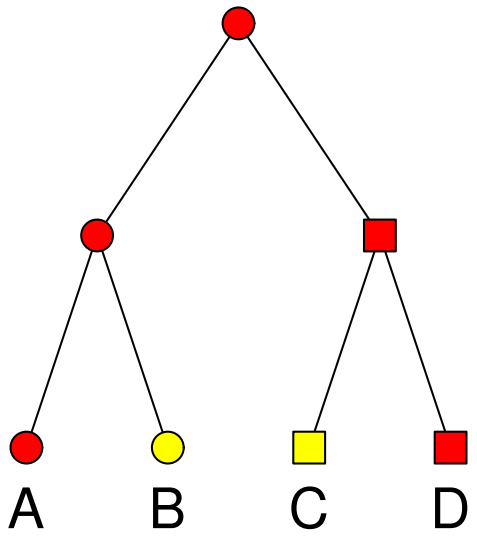
	def. art.	GenN	gender	tone	future	length
English	1	1	1	0	0	0
Spanish	1	0	1	0	1	0
Italian	1	0	1	0	1	1
Cantonese	0	1	0	1	0	1
Finnish	0	1	0	0	0	1

Reflection Exercise

What problems do you see with this method?

Problems, 1: Homoplasy

- Consider this tree, where each node carries two binary features (circle vs. square and red vs. yellow):



- The yellow value of the “colour feature” develops independently in two different lineages: **CONVERGENT EVOLUTION**. Such features are known as **HOMOPLASTIC**.

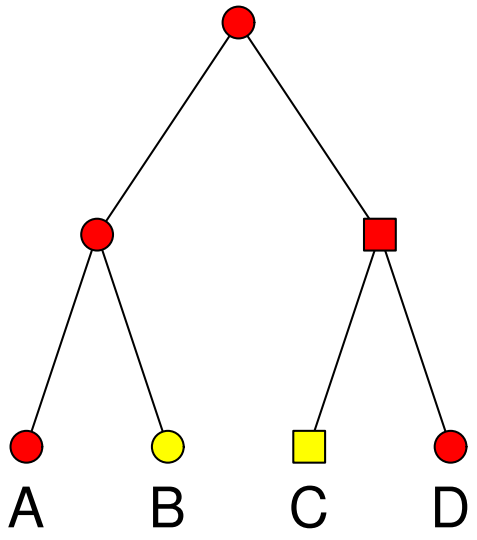
Problems, 1: Homoplasy

- Why is convergent evolution a problem for distance-based methods?
  - In the true evolutionary tree, two unrelated leaves can be highly similar
  - But a distance-based phylogenetic method equates similarity with relatedness, and so classifies the leaves wrongly
- Typological features are especially prone to homoplasy, since they are universal
- E.g. OV order can develop in any lineage — it's not the case that all OV languages are closely related!

  ⇒ We probably shouldn't use typological features in phylogenetic reconstruction…

Problems, 2: Back mutation

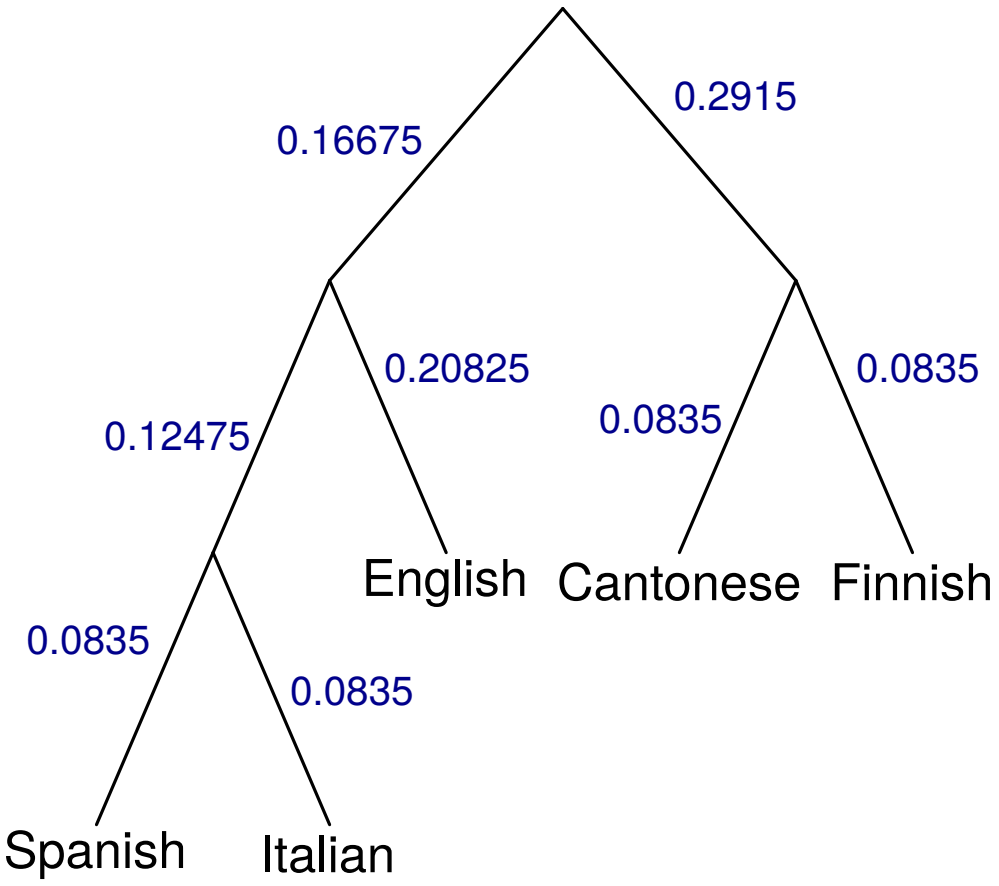
- Now consider this tree:



- Here there is a **BACK MUTATION** on the branch leading to D (circle > square > circle)
- As a result, A and D are identical
- Again a problem for distance-based methods

Problems, 3: Data sparsity

- Here's the metric tree we found last time:



Problems, 3: Data sparsity

- When applied to our (small!) data, UPGMA leads us to believe that
  - The branches leading to Cantonese and Finnish diverged not longer ago than Spanish and Italian
  - English diverged from the Romance languages longer ago than the “Cantonese–Finnish split”
- The problem is compounded by
  - the small number of features we considered (just 6)
  - the fact that the features were binary (less data resolution)

Problems, 4: The underlying model

- Three assumptions of UPGMA:
  1. Constant rate of evolution across lineages: Each leaf is at the same distance from the root
  2. Constant rate across features: Each feature evolves at the same rate
  3. Independence: Each feature evolves independently of each other feature
- (We called 1 the “molecular clock” last time)
- Each assumption is false, particularly for typological features
  1. Some lineages may change faster, e.g. because of geographical reasons
  2. Some features are known to change fast (e.g. vowels), some slow (e.g. word order)
  3. Implicational universals, e.g. a language with VSO order is unlikely to develop postpositions

Problems, 5: Borrowing

- **Borrowing** is defined as the passage of a feature from one branch of a tree to another due to language contact
- Then a tree representation is clearly inadequate to describe the historical passage from root to leaves
- This is a problem for **all** tree-based approaches, including but not limited to UPGMA

Addressing the problems

- Some of these problems can be addressed
- We can use more features (better data resolution)
- We can use features which are less prone to homoplasy and back mutation
- We can use features which are more likely to evolve with a molecular clock
- We can use features which are resistant to borrowing
- We can look for better tree-building algorithms, including different notions of linguistic distance

Part II

Lexical comparison methods (still distance-based)

Lexical items

- One option is to use lexical items instead of typological features
- Intuition: it is simply very unlikely that the same phonological form is used to encode the same meaning in different lineages (no homoplasy)
- However, “mutations” that occur within a given lineage may serve as a basis for subgroupings:

  Sp. /kabo/ Pt. /kabo/ It. /kapo/

- Directionality of sound change may guard against back mutations to some extent
- And we can try and pick characters which are resistant to borrowing

Swadesh list

I, you, we, this, that, who, what, not, all, many, one, two, big, long, small, woman, man, person, fish, bird, dog, louse, tree, seed, leaf, root, bark, skin, flesh, blood, bone, grease, egg, horn, tail, feather, hair, head, ear, eye, nose, mouth, tooth, tongue, claw, foot, knee, hand, belly, neck, breasts, heart, liver, drink, eat, bite, see, hear, know, sleep, die, kill, swim, fly, walk, come, lie, sit, stand, give, say, sun, moon, star, water, rain, stone, sand, earth, cloud, smoke, fire, ash, burn, path, mountain, red, green, yellow, white, black, night, hot, cold, full, new, good, round, dry, name

Swadesh list

- Compiled by linguist Morris Swadesh
- 100 concepts which are claimed to be
  - lexicalized in all languages
  - especially resistant against borrowing
- (Longer and shorter versions of the list also exist)
- Idea: use these 100 lexical items in tree reconstruction
- Assumption: two languages are the closer related the more their Swadesh words resemble each other
- Obvious but important point: we need to compare the phonological forms of the words, not their written forms!

Swadesh list

- To do this, we need a new notion of linguistic distance
- For two words with an equal number of phonemes, we could conceivably just take the Hamming distance as before:

$$H(\text{kabo}, \text{kapo}) = 1$$

- But what to do when the words are of different lengths?

$$H(\text{kabo}, \text{kap}) = ???$$

(/kap/ is the French reflex of Latin /kaput/)

- In that case, the Hamming distance is not defined

Levenshtein distance

- The **Levenshtein distance** between two words x and y, $L(x, y)$, is the **minimum** number of edits that turns x into y (or y into x)
- Possible edits are:
  - insertion of a phoneme
  - deletion of a phoneme
  - substitution of a phoneme
- Examples:
  - $L(\text{kabo}, \text{kapo}) = 1$ (one substitution)
  - $L(\text{kapo}, \text{kap}) = 1$ (one deletion)
  - $L(\text{kabo}, \text{kap}) = 2$ (one substitution and one deletion)
- Exercise: compute
  - $L(\text{kabo}, \text{hɛd})$
  - $L(\text{kabo}, \text{kɔpf})$
  - $L(\text{kapo}, \text{kɔpf})$

Levenshtein distance

- $L(x, y)$ gives the distance between two words
- To get the distance between two *languages*, we sum over all words in the Swadesh list
- Let $x_{i,A}$ denote the ith word in the list in language A
- Then the distance of two languages A and B is

$$d(A, B) = \sum_i L(x_{i,A}, x_{i,B})$$

- We can also divide by the number of words in the list to get a normalized measure. For a 100-word list:

$$\overline{d}(A, B) = \frac{1}{100} \sum_i L(x_{i,A}, x_{i,B})$$

Levenshtein distance in R

- To compute the Levenshtein distance in R, use the built-in `adist` function:

```
x <- "kapo"
y <- "kabo"
z <- "kopf"
w <- "hed"
adist(x, y)
adist(x, z)
adist(z, w)
adist(y, z)
```

Length correction

- Compare:
  - Sp. *es* /es/ and It. *è* /ɛ/ ‘he/she/it is’
  - Sp. *proporcionado* /proporsjonado/ and It. *proporzionato* /proportsjonato/ ‘proportionate’
- Levenshtein distances:
  - $L(\text{es}, \varepsilon) = 2$
  - $L(\text{proporsjonado}, \text{proportsjonato}) = 2$
- However, in some sense the longer words are more similar to each other than the shorter words
- I.e. a couple of edits make more of a difference when the word is short
- Solution: correct by dividing with the number of phonemes in the longer word
- For example:
  - $L_C(\text{es}, \varepsilon) = 2/2 = 1$
  - $L_C(\text{proporsjonado}, \text{proportsjonato}) = 2/14 = 1/7 \approx 0.14$

Length correction

- In R,
  - the `nchar` function returns the number of characters in a string
  - the `max` function returns the greatest of its arguments
- So, we can get the length-corrected Levenshtein distance of two strings as follows:

```
x <- "es"
y <- "E"
z <- "proporsjonado"
w <- "proportsjonato"
adist(x,y)/max(nchar(x), nchar(y))
adist(z,w)/max(nchar(z), nchar(w))
```

Length correction

- It makes sense to write this up as a function
- Type the following using a text editor:

```
adist_corr <- function(x, y) {
  adist(x,y)/max(nchar(x), nchar(y))
}
```

- Save the file as `adist_corr.R` in your R working directory
- Load the function into your R session:

```
source("adist_corr.R")
```

- Now you can simply type:

```
adist_corr(x,y)
```

Distance matrices

- To get the Levenshtein distance between two languages, we need to sum over all items in the Swadesh list
- I have written a function that does this and produces a distance matrix as output
- Go to ILIAS and download `ldist.R` from the Code folder
- Put the file in your R working directory and load it:

```
source("ldist.R")
```

Distance matrices

- Let's construct a dummy lexical matrix:

```
Spanish <- c("kabo", "kabra", "persona")
Portuguese <- c("kabo", "kabra", "pes:oa")
German <- c("kopf", "tsig@", "perzon")
lexical_mtx <- rbind(Spanish, Portuguese, German)
lexical_mtx
```

```
##            [,1]   [,2]    [,3]
## Spanish    "kabo" "kabra" "persona"
## Portuguese "kabo" "kabra" "pes:oa"
## German     "kopf" "tsig@" "perzon"
```

Distance matrices

- And apply the `ldist` function to produce a Levenshtein distance matrix:

```
ldist(lexical_mtx)

##                Spanish Portuguese German
## Spanish             NA          NA     NA
## Portuguese           3          NA     NA
## German              10          11     NA
```

Distance matrices

- The `ldist` function takes two optional arguments
- `normalize` may be used to normalize the distances by the number of lexical items in the lexical matrix:

```
ldist(lexical_mtx, normalize=TRUE)
```

```
##                Spanish Portuguese German
## Spanish             NA          NA     NA
## Portuguese    1.000000          NA     NA
## German        3.333333    3.666667     NA
```

- (The default value is `normalize=FALSE`)

Distance matrices

- `correct` gives length-normalized distances:

```
ldist(lexical_mtx, correct=TRUE)

##                Spanish Portuguese German
## Spanish             NA          NA     NA
## Portuguese 0.4285714            NA     NA
## German     2.0357143          2.25     NA
```

- (The default value is `correct=FALSE`)

Distance matrices

- The two options can be combined:

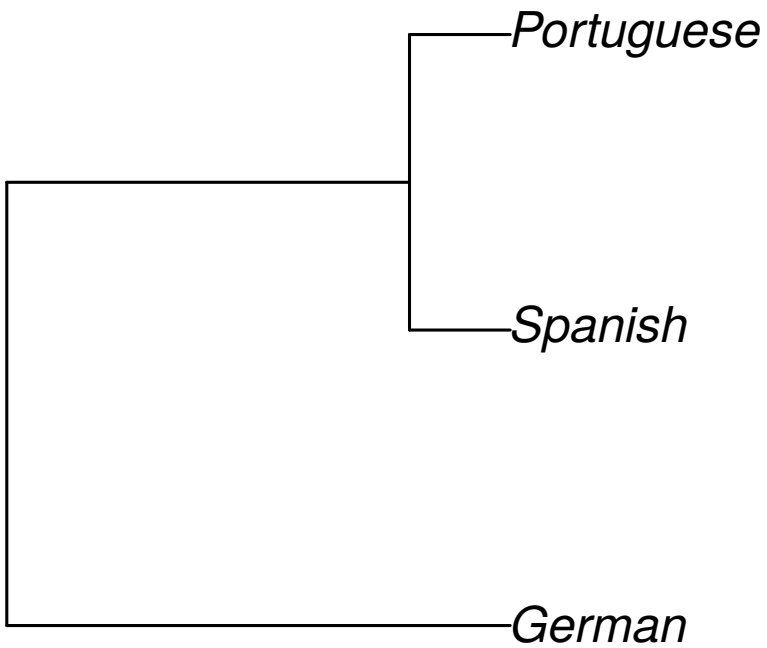
```
ldist(lexical_mtx, normalize=TRUE, correct=TRUE)

##                Spanish Portuguese German
## Spanish             NA          NA     NA
## Portuguese 0.1428571            NA     NA
## German     0.6785714          0.75     NA
```

Distance matrices

- We can now apply UPGMA to infer a tree from the distance matrix:

```
library(phangorn)
distance_mtx <- ldist(lexical_mtx, normalize=TRUE,
                      correct=TRUE)
tree <- upgma(distance_mtx)
plot(tree)
```

Distance matrices



An aside: how to export graphics from R?

- We've been drawing trees in R for a while now
- Sometimes, it would be useful to save those images so that they can be included in a word-processor document (e.g. for the purposes of your Portfolio)
- Here's how to produce a PNG file:

```
png("mypicture.png", width=600, height=600)
plot(tree)
dev.off()
```

- First, the `png` command opens a *graphics device*. Then we plot the tree. Finally, we close the graphics device (`dev.off`), and the picture is actually written into the file (`mypicture.png`).

An aside: how to export graphics from R?

- Other graphics file formats are also available. For example, you can produce PDF graphics as follows:

```
pdf("mypicture.pdf", width=5, height=6)
plot(tree)
dev.off()
```

- Both the `png` and `pdf` commands can take many optional arguments to specify things such as font, font size and so on. See the help pages:

```
?png
?pdf
```

Part III

Real data

The ASJP 40-item list

- The ASJP (Automated Similarity Judgment Program) database (`http://asjp.clld.org/`) employs a 40-item subset of the Swadesh list:

I, you, we, one, two, person, fish, dog, louse, tree, leaf, skin, blood, bone, horn, ear, eye, nose, tooth, tongue, knee, hand, breast, liver, drink, see, hear, die, come, sun, star, water, stone, fire, path, mountain, night, full, new, name

- This list is given for 7655 languages (!)
- (The full Swadesh list is given by ASJP for a subset of these languages)

The ASJP 40-item list

- Download the ASJP 40-item data from ILIAS (`asjp-small.csv` in the Data folder)
- And read the CSV file into your R session:

```
asjp <- read.csv("asjp-small.csv", row.names=1)
```

- Don't forget the `row.names=1` argument
- (It specifies that the row names of the data frame are to be taken from the first column of the CSV)

The ASJP 40-item list

- Show the first 5 lines of the data frame to get an idea of how the data is structured:

```
head(asjp)
```

- Or try:

```
View(asjp)
```

- So, each row represents one language, and the columns are the 40 lexical items

The ASJP 40-item list

- The entire data frame contains a *lot* of data:

```
nrow(asjp)

## [1] 7655
```

- Let's subset it and work with a small sample of languages
- To get just one language, you specify the relevant row name, for example:

```
asjp["ENGLISH",]
```

- (ASJP language names are in ALL-CAPITALS)

The ASJP 40-item list

- To see which languages are available, you'll have to examine:

```
rownames(asjp)
```

- To subset a number of languages, use the `c` vector constructor:

```
asjp[c("ENGLISH", "FRENCH"),]
asjp[c("ENGLISH", "FRENCH", "FINNISH", "URDU"),]
```

Exercise

1. Form a subset of the ASJP data consisting of the following languages: ENGLISH, FRENCH, FINNISH, HUNGARIAN, STANDARD_GERMAN, DUTCH, SPANISH, PORTUGUESE, ESTONIAN, URDU, BENGALI, GUJARATI, MANDARIN
2. Store this in a variable called `asjp_subset`
3. Use `ldist` to form a normalized and length-corrected Levenshtein distance matrix
4. Use `upgma` to infer a tree from the distance matrix
5. Use `plot` to draw a picture of the tree
6. Is the resulting phylogenetic tree reasonable based on what you know about the world's language families?

Next week

- Next week is the reading week
- No class, properly speaking
- *However*, the computer lab will be at your disposal on Wednesday between 15:15 and 17:45
- I will also be there, so that you may ask questions and get help with the Portfolio Exercises