

Tutorial accompanying the study "Computer-Assisted Language Comparison: State of the Art"

Wu, Mei-Shin; Schweikhard, Nathanael E.; Bodt, Timotheus A.; Hill, Nathan W.; List, Johann-Mattis

This tutorial supplements the study "Computer-Assisted Language Comparison: State of the Art". In this tutorial, we explain in detail, how our workflow can be tested and applied.

The workflow consists of several Python libraries that interact, one producing the data that can be used by the other. Since the data is available in different stages, each stage allows us to intervene by correcting errors manually that were made by the automated approach.

For users who are interested in testing our workflow on their local machine or further applying it in their own research, some basic knowledge of the Python programming language and the commandline will be required. All the software offered here is available in the form of free software. For more information on LingPy, the main programming library used here, we recommend users to check the tutorial¹ accompanying the study "Sequence comparison in computational historical linguistics"² by List et al. (2018)[1].

1. Code Ocean Capsule

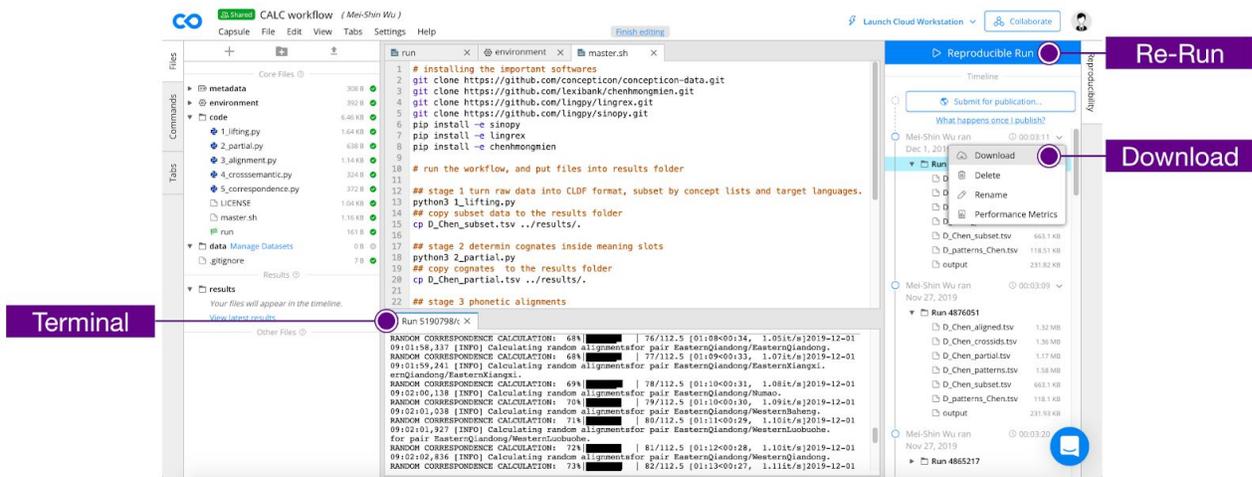
In order to facilitate it for users to quickly test our workflows without installing the software, we have set up a Code Ocean Capsule which users can use to run the code remotely. Code Ocean is an open access platform which enables researchers to reproduce their or others' experiments. For a detailed introduction to the Code Ocean platform³, please refer to the website. To see how our experiments can be run from within the Code Ocean Capsule, follow the following steps:

- a) Navigate to the capsule: <https://codeocean.com/capsule/8178287/tree/v2>
- b) Press the **Re-Run** button to reproduce the results.
- c) View the progression in the **Terminal** panel.
- d) Download all results and unzip the .zip file for further inspection on EDICTOR.

¹ <https://github.com/lingpy/lingpy-tutorial>

² <https://academic.oup.com/jole/article/3/2/130/5050100>

³ <https://codeocean.com/>



The following files can be found in the downloaded file:

File	Stage	Section
D_Chen_subset.tsv	From raw data to tokenized data	3.1
D_Chen_partial.tsv	From Tokenized Data to Cognate Sets	3.2
D_Chen_aligned.tsv	From Cognate Sets to Alignments	3.3
D_Chen_crosssids.tsv	From Alignments to Cross-Semantic Cognates	3.4
D_Chen_patterns.tsv	From Cross-Semantic Cognates to Sound Correspondence	3.5
D_Chen_distance.dst	Validation	4.2, 4.3
D_Chen_tree.tre	Validation	4.2, 4.3

2. Installation Instructions

We assume that users who are interested in running the workflow on their local machine are familiar with the essentials of command-line operations and system administration on either Unix-like systems (such as Linux and MacOS) or Windows systems. Also, users should have Python⁴ installed, including the package manager `pip`. Additionally, the version control system⁵

⁴ <https://www.python.org/>, Version 3.5 or higher

⁵ <https://git-scm.com/>

git will be required. We strongly encourage users to run this code in a virtual environment. A virtual environment is a practical solution for creating independent configurations for testing and experimenting, with no interference on the system-wide installation and without requiring complex virtualization or containerization solutions. The Python Packaging User Guide⁶ gives clear instructions on setting up a virtual environment on Windows, Linux and macOS.

We start by installing the dependencies from the commandline. In order to do so, we first download the code that we will use with help of git.

```
$ git clone https://github.com/lingpy/workflow-paper.git
$ cd workflow-paper
```

Now that we have done this, we can install all the packages we will need with help of pip.

```
$ pip install -r requirements.txt
```

Now that this has been done, we need to configure the access to reference catalogs, such as Concepticon⁷ and CLTS⁸ in order to make sure that they can be accessed readily by the code. This can be done with help of the `catconfig` argument submitted with the `cldfbench` package which organizes the linguistic datasets.

```
$ cldfbench catconfig
```

You will be prompted to ask if you want to clone actual versions of Concepticon, Glottolog, and CLTS, and the easiest way to deal with this is to agree and type “y” in all cases.

3. Getting Started

There are two basic ways in which you can run our workflow:

1. You can run it by downloading a set of Python scripts and running them directly on your computer.

⁶ <https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/>

⁷ <https://github.com/concepticon/concepticon-data>

⁸ <https://github.com/cldf-clts/clts/>

2. You can use the `cldfbench` package to run the commands via the commandline, without downloading the data directly.

The advantage of solution 2 is that you do not have to download extra data, since we have integrated the code directly in the `lexibank` version of the dataset of Hmong Mien languages by Chén (2012)[2]. Once this dataset has been installed (and this is the first package we have installed in the previous section as part of all dependencies needed), you can type commands on your commandline, and the code will be carried out. The disadvantage is that the code example itself is not that easy to process for people less experienced with Python. For this reason, we will only note the commands in each of the steps we discuss in the following, and not explain them in more detail.

3.1 From Raw Data to Tokenized Data

The first script essentially loads the data from the repository and creates a wordlist that contains a subselection of all the data that was used. Some aspects of the more difficult “lifting” of data have already been done and distributed along with the original data package⁹, which specifically also contains the orthography profile in the file `etc/orthography.tsv` and can be automatically applied with help of the `cldfbench` package.

```
$ cldfbench lexibank.makecldf chenhmongmien
```

But since the data is available in the form of a `cldf` package with the original orthography already tokenized to the formats we need, you can also skip this step and convert the data to the wordlist format required by the `lingpy` package.

```
$ python 1_select.py
```

If you want to test the version from the CLDF-repository directly with `cldfbench`, you can type:

```
$ cldfbench chenhmongmien.wf_select.
```

This will select a part of the languages and a part of the concepts, as indicated in the main study and write them to a file `D_Chen_subsets.tsv`. Additionally, you will see some statistics on the

⁹ <https://github.com/lexibank/chenhmongmien>

terminal, specifically a table indicating the coverage for each language. If you want to select all languages, and not just a subset, type:

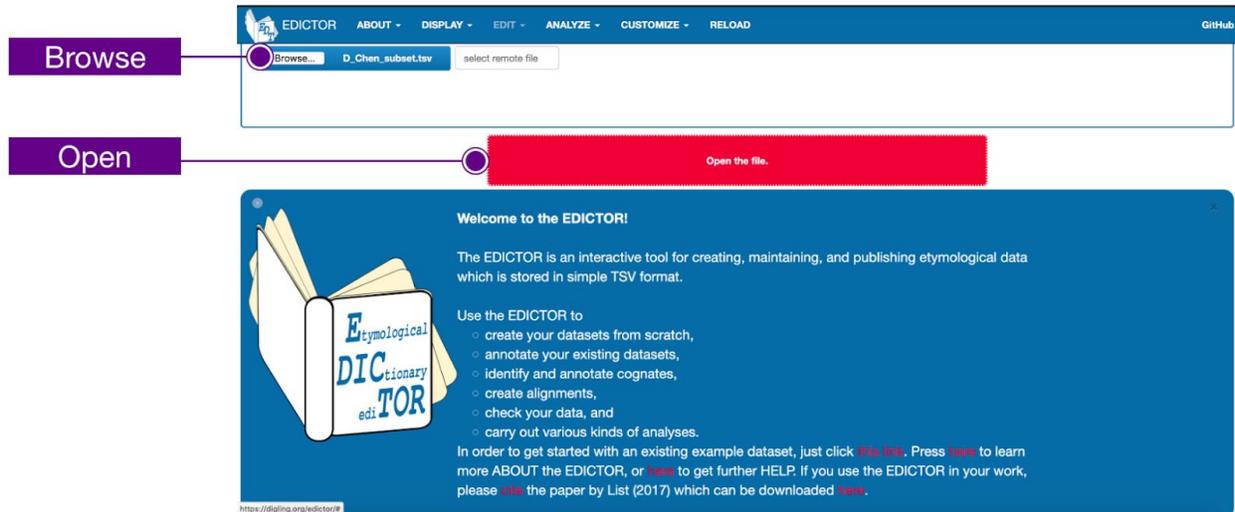
```
$ python 1_select.py all
```

The output `A_Chen_subset.tsv` is generated due to the argument `all` is used. Once the argument `all` is used in the first stage, it has to be added to the rest of stages to ensure that the workflows process the correct files.

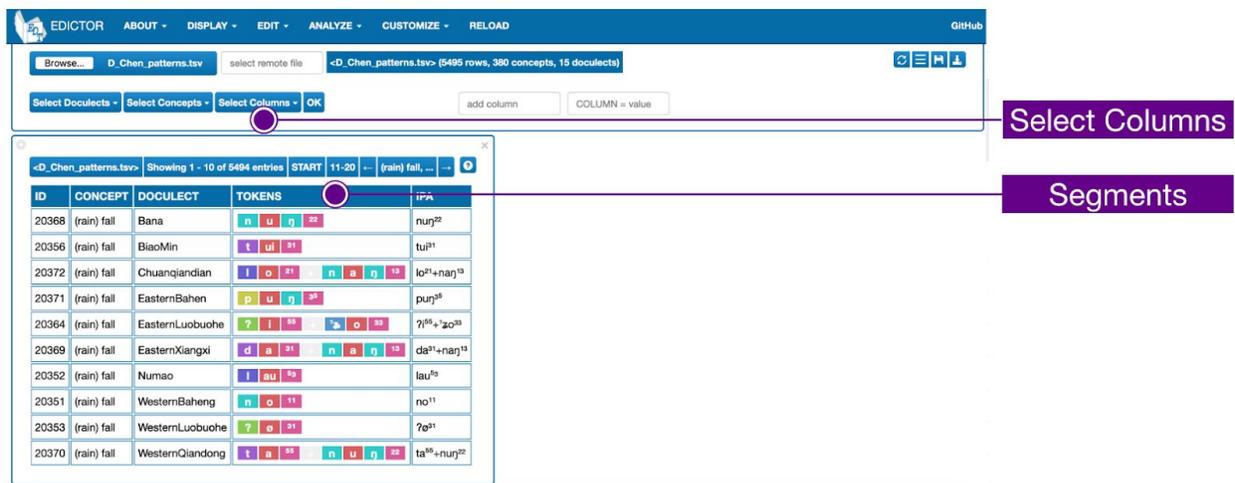
Doculect	Words	Coverage
Bana	502	1.00
BiaoMin	488	0.97
CentralGuizhouChuanqiandian	454	0.90
Chuanqiandian	501	1.00
EasternBahen	492	0.98
EasternLuobuohe	499	0.99
EasternQiandong	442	0.88
EasternXiangxi	492	0.98
Numao	490	0.98
WesternBaheng	500	1.00
WesternLuobuohe	488	0.97
WesternQiandong	494	0.98
WesternXiangxi	502	1.00
Younuo	500	1.00
ZaoMin	455	0.91

Already now you can inspect the data with the help of the [EDICTOR](#) tool. In order to do so, open the tool's website at <https://digling.org/edictor/> and wait until the page is loaded (note that we recommend to browse EDICTOR in Firefox, but GoogleChrome should also not cause further problems).

The data is in the file `D_Chen_subset.tsv`, in order to load it to the tool, press the **Browse** button and select the file. Once this has been done, press the **Open the file** button to examine the data, as illustrated in the following figure.

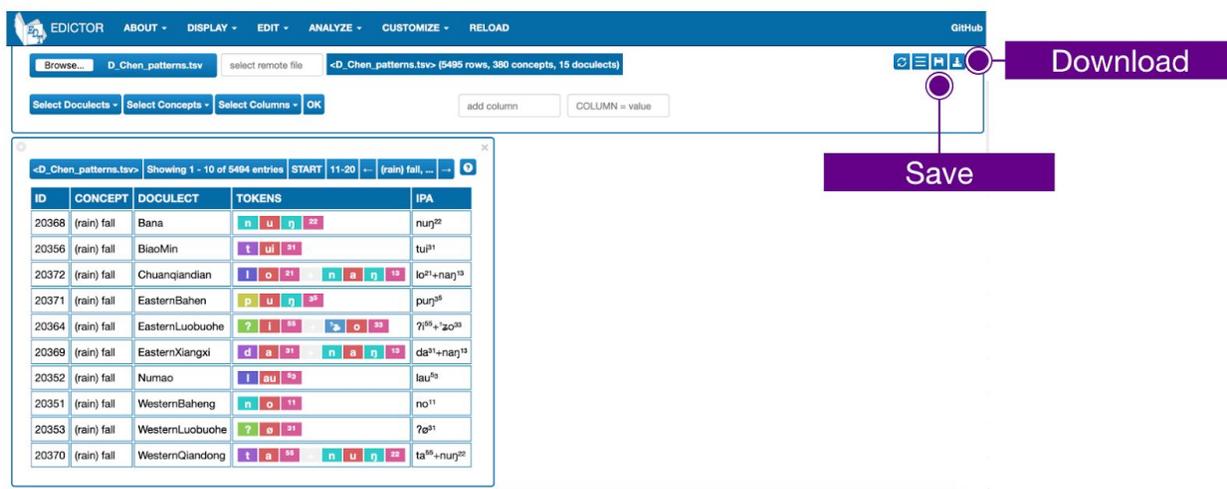


The segmented strings are displayed in the **TOKENS** column. Press **Select Columns** to inspect the raw forms and other aspects of the data, as shown in the following figure.



In order to save data to your computer, after you have manually edited them, you need to “download” them. This may be a bit surprising, since effectively, you do not download the data, but since the EDICTOR is working on a browser, it does not have any access to the data on your computer, and **download** is the only way to communicate with your machine. Thus, in order to save your data and load it to your machine, you first have to press the **save** icon at the top-right corner in order to store the edited data in the web browser. When now pressing the

download icon at the top-right, your browser will either directly download the data and store them in your download folder, or it will ask you to specify a specific file destination.



The screenshot shows the EDICTOR web interface. At the top, there is a navigation bar with options: EDICTOR, ABOUT, DISPLAY, EDIT, ANALYZE, CUSTOMIZE, and RELOAD. A GitHub logo is in the top right. Below the navigation bar, there is a file browser area with a 'Browse...' button and a text input field containing '<D_Chen_patterns.tsv>'. To the right of this input are icons for refresh, list, and download. A purple callout box labeled 'Download' points to the download icon. Below the file browser, there are buttons for 'Select Doculects -', 'Select Concepts -', and 'Select Columns -', followed by an 'OK' button. There is also an 'add column' button and a 'COLUMN = value' input field. The main content area displays a table with the following data:

ID	CONCEPT	DOCULECT	TOKENS	IPA
20368	(rain) fall	Bana	n u ŋ ʔ	nur ²²
20356	(rain) fall	BiaoMin	t u i ʔ	tu ²¹
20372	(rain) fall	Chuangqian	l o ʔ n a ŋ ʔ	lo ²¹ +nar ¹³
20371	(rain) fall	EasternBahan	p u ŋ ʔ	pur ²¹
20364	(rain) fall	EasternLuobuohe	ʔ i ʔ ʔ ʔ ʔ ʔ ʔ	ʔi ²² +ʔo ²²
20369	(rain) fall	EasternXiangxi	d a ʔ n a ŋ ʔ	da ²¹ +nar ¹³
20352	(rain) fall	Numao	l au ʔ	lau ²⁵
20351	(rain) fall	WesternBaheng	n o ʔ	no ¹¹
20353	(rain) fall	WesternLuobuohe	ʔ o ʔ	ʔo ²¹
20370	(rain) fall	WesternQiandong	t a ʔ n u ŋ ʔ	ta ²⁰ +nur ²²

Be careful when editing data in the EDICTOR without saving and downloading them. If you close your browser, all the edits you made will be lost, so you should regularly save and download your data when working with the EDICTOR. As a shortcut, you can also type CONTROL+S to save and CONTROL+E to “export” the data (i.e., to download them).

3.2 From Tokenized Data to Cognate Sets

Partial cognate detection is an important task, specifically when working with Southeast Asian language data. The algorithm we use for this task was first proposed in the study “Using Sequence Similarity Networks to Identify Partial Cognates in Multilingual Wordlists” by List et al. (2016)[3], where the algorithm is described in due detail.

To illustrate how the algorithm works, we provide an example with four words for ‘moon’ in the Eastern Baheng, Eastern Qiandong, Bana and Biao Min language varieties.

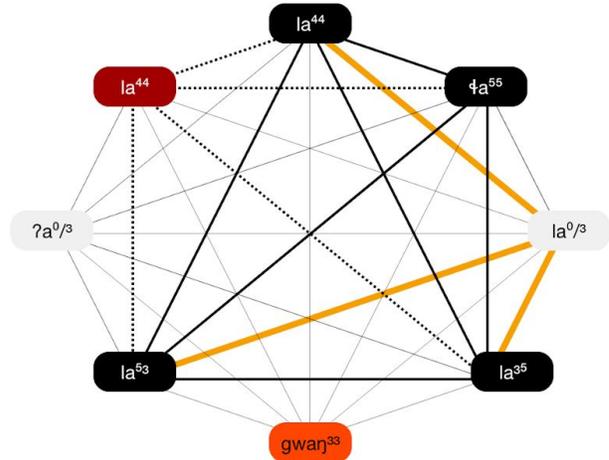
The major steps of the algorithm are the following:

1. Calculate the distances of all morpheme pairs.
2. Create a fully connected network from the distance scores.
3. Filter the network by deleting edges in the following fashion:
 - A. Two morphemes in the same word should not be linked (see the dashed lines in the following figure).

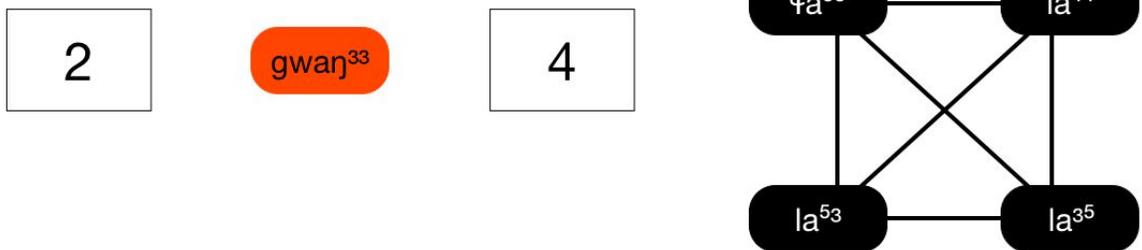
B. A morpheme in a word should not be linked to two morphemes in another word (see the yellow edges in the figure).

4. Remove the edges with similarity scores below a given threshold.

DOCULECT	IPA
EasternBaheng	la ^{0/3} ɬa ⁵⁵
EasternQiandong	la ⁴⁴ la ⁴⁴
Bana	la ^{0/4} la ³⁵
BiaoMin	la ⁵³ gwan ³³



Once this has been done, an algorithm for Community Detection in networks[4] is used to partition the network into “communities”, with each community representing one partial cognate set.



In order to calculate partial cognates, we use the algorithm as provided by the `lingpy` software package and apply it to our subselection of languages.

```
$ python 2_partial.py
```

If you want to test the version from the CLDF-repository directly with `cldfbench`, you can type:

```
$ cldfbench chenhmongmien.wf_partial.
```

This will take some time when you run it the first time. The data can be found in the file `D_Chen_partial.tsv`.

To inspect the data with EDICTOR, load `D_Chen_partial.tsv` as shown before. Then press **DISPLAY** to select **SETTINGS** in the drop-down menu. Select **PARTIAL** in the **Morphology and Colexification Mode** entry. Press the **Refresh** button.

The screenshot shows the EDICTOR web interface. On the left, a table displays data for 10 entries from the file `<D_Chen_patterns.tsv>`. The table has columns for ID, CONCEPT, DOCULECT, TOKENS, IPA, and COGIDS. The COGIDS column contains numbers ranging from 1 to 8. On the right, the 'Settings of the Editor' panel is open, showing options for Preview (10), Cognate IDs (PARAMETERID), Partial Cognates (COGIDS), Tokens (tokens column), Alignments (ALIGNMENT), and Morphology and Colexification Mode (PARTIAL selected). Three purple callout boxes with white text point to specific elements: 'Display' points to the top navigation bar, 'Refresh' points to a circular refresh icon in the settings panel, and 'Partial' points to the 'PARTIAL' radio button in the Morphology and Colexification Mode section.

ID	CONCEPT	DOCULECT	TOKENS	IPA	COGIDS
20368	(rain) fall	Bana	n u ŋ ʔ	nɯŋ ²²	2
20356	(rain) fall	BiaoMin	t u ʔ	tɯ ²¹	3
20372	(rain) fall	Chuanqiandian	l o ʔ n a ŋ ʔ	lo ²¹ +nan ¹³	1.2
20371	(rain) fall	EasternBahen	p u ŋ ʔ	pɯŋ ²⁵	5
20364	(rain) fall	EasternLuobuohu	ʔ ²⁵ +ʔo ²³	ʔ ²⁵ +ʔo ²³	7.6
20369	(rain) fall	EasternXiangxi	d a ʔ n a ŋ ʔ	da ²¹ +nan ¹³	3.2
20352	(rain) fall	Numao	l a u ʔ	lau ²³	1
20351	(rain) fall	WesternBaheng	n o ʔ	no ¹¹	8
20353	(rain) fall	WesternLuobuohu	ʔ a ʔ	ʔa ²¹	6
20370	(rain) fall	WesternQiandong	t a ʔ n u ŋ ʔ	ta ²⁵ +nɯŋ ²²	3.2

In order to investigate the partial cognates, you need to select the column which stores the identifiers. To do so, press **Select Columns** and select **COGIDS** in the drop-down menu.

If you right-click on any number in the “COGIDS” column, a pop-up window will open and show all the cognate sets for a given word form in the form of an alignment. Since we have not yet aligned the data, the alignment will be wrong at this point.

The screenshot shows the EDICTOR interface. The main window displays a table with columns: ID, CONCEPT, DOCULECT, TOKENS, IPA, and COGIDS. A right-click context menu is open over the 'TOKENS' column of the first row, with an arrow pointing to it labeled 'Right click!'. A secondary window titled 'Bana -sun (COGIDS: 6542,6541)' is open, showing a detailed view of the 'Bana' entry with a grid of colored annotations over the tokens, with an arrow pointing to it labeled 'Alignment'.

ID	CONCEPT	DOCULECT	TOKENS	IPA	COGIDS
31	sun	Bana	l a n i	la ⁷ +ni ¹³	6542 6541
43	sun	BiaoMin	p i t au	pi ²¹ +tau ³¹	6541 6548
49	sun	CentralGuizhouChuanqiandian	o e s	o ²⁴	6541
40	sun	Chuanqiandian	o o s	o ²⁵	6541
28	sun	EasternBahan	l a n o e s	la ⁷ +o ²⁵	6542 6541
44	sun	EasternLuobuohe	q o n a s i	qo ⁷ +na ³¹	6547 6541
47	sun	EasternQiandong	q a n ei s	q'an ²⁵ +nei ²⁴	6546 6541
33	sun	EasternXiangxi	q a n ei s	q'a ²⁵ +nei ²⁴	6543 6541
48	sun	Numao	m a n a i s	ma ²² +na ³¹	6545 6541
29	sun	WesternBaheng	q o n i a n ei s	q'on ³¹ +?an ²⁵ +pei ²⁸	6546 6550 6541

3.3 From Cognate Sets to Alignments

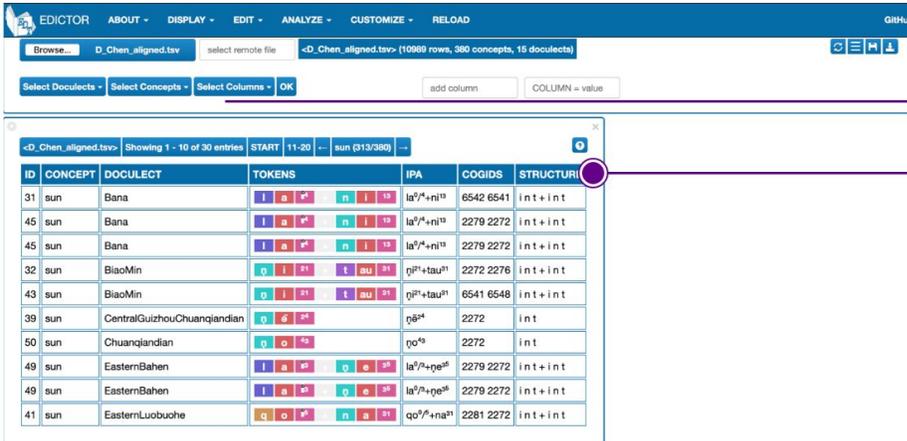
To align the data, we use the new procedure for template-based alignment, which is available from the `lingrex` package which we have installed as one of the requirements of our workflow, and the `sinopy` package, which helps us to compute syllable templates from all morphemes in the data. Running the code is again straightforward.

```
$ python 3_alignment.py
```

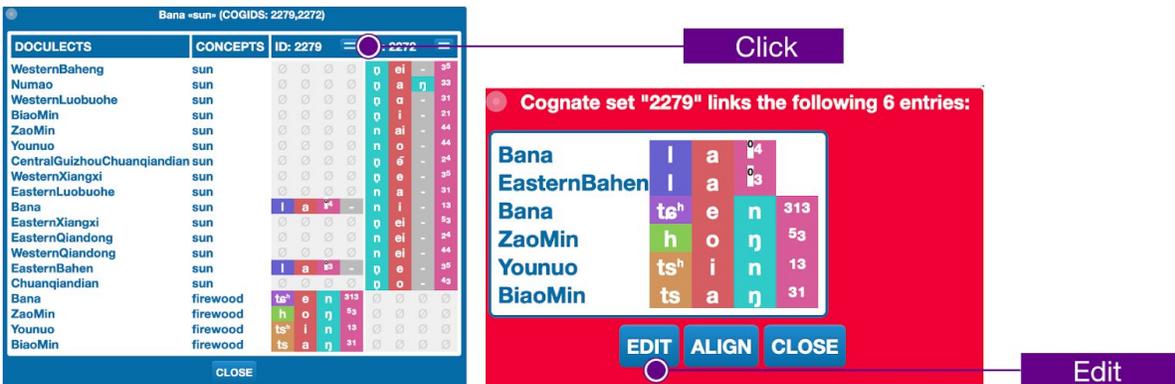
If you want to test the version from the CLDF-repository directly with `cldfbench`, you can type:

```
$ cldfbench chenhmongmien.wf_alignment
```

The aligned data will be stored in the file `D_Chen_aligned.tsv`. To inspect the alignments in EDICTOR, load this file and follow the previous steps we mentioned in Section 3.2. In addition to selecting the **COGIDS** column now, we also select the **STRUCTURE** column, since this column provides the templates for each morpheme, which we have automatically added to the data with help of `sinopy`.



As we already mentioned, if you right-click on any number in the “COGIDS” column, a pop-up window will show the alignment. Click on the = sign to modify the alignment. The modification itself is very straightforward: just click on a sound segment to move it to the right, and click on a gap segment to delete this segment.



3.4 From Alignments to Cross-Semantic Cognates

The algorithm for cross-semantic cognate detection as we propose it here is illustrated in more detail in the main study. It is implemented as part of the `lingrex` package. Again, it is straightforward to run the code.

```
$ python 4_crosssemantic.py
```

If you want to test the version from the CLDF-repository directly with `cldfbench`, you can type:

```
$ cldfbench chenhmongmien.wf_crosssemantic
```

The output file is `D_Chen_crossids.tsv`, and we load it into the EDICTOR tool, just as we did before, but when checking the **SETTINGS** in the menu this time, we need to specify that the column “CROSSIDS” holds the partial cognates. To do so, just type in **CROSSIDS** in the text field **Partial Cognates** in the settings menu and then press the **refresh** button.

The screenshot shows the EDICTOR interface with the 'Settings of the Editor' panel open. The 'Partial Cognates' field is set to 'CROSSIDS'. The 'Morphology and Colexification Mode' is set to 'PARTIAL'. A 'Refresh' button is visible. Callouts point to these elements.

ID	CONCEPT	DOCULECT	TOKENS	CROSSIDS
45	sun	Bana	l a s n l s	2279 2272
32	sun	BiaoMin	q i s t a u s i	2272 2276
39	sun	CentralGuizhouChuanqiandian	q e s	2272
50	sun	Chuanqiandian	q o s	2272
49	sun	EasternBahen	l a s q e s	2279 2272
41	sun	EasternLuobuohu	q o s n a s i	2281 2272
47	sun	EasternQiandong	q a n s n e i s	2274 2272
46	sun	EasternXiangxi	q a s q e i s	2277 2272
28	sun	Numao	m a s q a n s	2280 2272
27	sun	WesternBaheng	e i o s i a n s q e i s	2274 2275 2272

To inspect the distribution of partial cognates, press **ANALYZE** in the top-level menu and select **Cognate sets** in the drop-down menu.

The screenshot shows the EDICTOR interface with the 'ANALYZE' menu open and the 'Cognate Sets' panel selected. The 'Cognate Sets' panel shows a list of cognate sets with columns for ID, CONCEPT, DOCULECT, TOKENS, and IPA. Callouts point to the 'ANALYZE' menu and the 'Cognate Sets' panel.

ID	CONCEPT	DOCULECT	TOKENS	IPA
7646	house	Bana	l a s n l s	la ⁵⁵ +pja ⁴⁴
7658	house	BiaoMin	q i s t a u s i	pi ⁴⁵
7664	house	CentralGuizhouChuanqia...	q e s	pk ¹³
7655	house	Chuanqiandian	q o s	tse ⁵⁵
7643	house	EasternBahen	l a s q e s	qj ⁵⁵
7659	house	EasternLuobuohu	q o s n a s i	qo ⁷⁷ +pai ⁵⁵
7662	house	EasternQiandong	q a n s n e i s	tse ⁵⁵
7648	house	EasternXiangxi	q a s q e i s	pu ⁵⁵
7663	house	Numao	m a s q a n s	pje ⁵⁵
7644	house	WesternBaheng	e i o s i a n s q e i s	ʔa ⁷⁷ +tjo ⁵⁵
7660	house	WesternLuobuohu	q o s n a s i	ʔa ⁷⁷ +pjo ⁵⁵
7642	house	WesternQiandong	q a n s n e i s	pe ⁵⁵
7649	house	WesternXiangxi	q a s q e i s	pzu ⁴⁴
7657	house	Youmao	q u s	pu ⁵⁵
7652	house	ZaoMin	q i s t a u s i	pju ⁴⁴

As a result, a new panel will open and show the distribution of all cognate sets across the different language varieties. Pressing the red button with the cognate set identifier on the left will open the alignment. Pressing the yellow buttons with the word identifiers will show you the original morpheme. On the right, in the column **CONCEPTS**, you will find those cognate sets which are attested for more than one concept as separated by a comma. Clicking on this field will modify the main wordlist panel in such a way that only the selected concepts will appear.

Investigate cognate sets in the data

Select Sets: [OK] [130 of 14 Sets]

CROSSIDS	Wes	Eas	Wsa	Ban	Eas	Wsa	Zao	Chu	You	Bis	Eas	Wsa	Hum	Eas	Can	CONCEPTS	
112	209	251			2075	2032	2026	1441	2085	3144	307	367		1448	171	wash,back,branch,leaf,root	
241	2334	2933			2940	2941	2944	2947	2950	2951	2952	2955	2954	2956		blood	
281	4925	4926	4927	4929	4931	4932		4938	4940		4942	4943	4946	4945	4947	branch/ twig	
388	18655	19291	19292	19294		18662	19300	18668		19306	18672	18673	18676	18675	18677	climb (a tree),go upstairs	
481	18124		18128	18130	18131	18134	18137	18139	18140	18141	18142			18144	18146	cry	
575	17303	17304	17305	17307		17310	17313	17316	17318		17320	17321	17324	17323	17325	drink	
644	10529	10530		10533	10535	0756	10539	10542		10545	10546	10547	10550	10549	10551	eight,egg	
681			11413	11415	11417	11418	11421	11424	11426	11427	11428	11429	11432	11431	11433	everyday	
893	2089	2090	2091		2095		2099	2102	2104	2105	2106	2107	2110	2108	2111	eye	
724	6551	6552	6553	6555	6557	6558	6561	6564	6566		6568	6569	6571		6572	fat meat	
830	16750	16751	16752	16754	16756	16757	16759			16767	16768	16771	16770	16772		fly	
901	5048	5049	5050	5052	5054	5055	5058		5063	5064	5065	5066	5069		5070	fruit	
962	14439	14440	14441		14445	14446	14449	14452		14455	14456	14457	14460	14459	14461	good	
2447	7642	7643		7648	7648	7649	7652		7657	7658	7659	7660	7663	7710	7664	house,thatched cottage,ill...	
1172	956	957	958	960	962	963		969	971		973	974	977	976	978	human being	
1191	20894	20895	20896	20898	20900	20901	20904	20907	20909	20910	20911			20914	20916	illuminate	
1282	20992	20993	20994	20996	20998		21002	21005	21007	21008	21009			21013	21012	21014	know
1294	20448	20447	20448		20452	20453	20456	20459	20461		20463	20464	20467	20466	20468	laugh (smile)	

3.5 From Cross-Semantic Cognates to Sound Correspondence Patterns

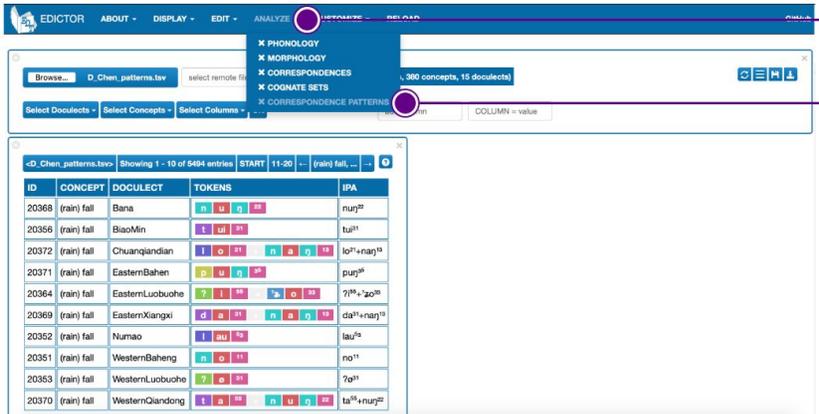
As a final step, we will try to infer the major correspondence patterns in the data, using the algorithm by List (2019)[5] which is available from the `lingrex` package. Running the code is straightforward, as before.

```
$ python 5_correspondence.py
```

If you want to test the version from the CLDF-repository directly with `cldfbench`, you can type:

```
$ cldfbench chenhmongmien.wf_correspondence
```

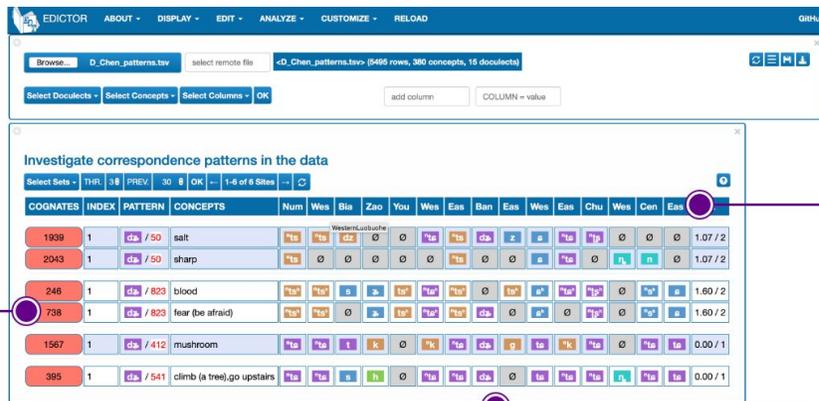
This creates two output files. One, called `D_Chen_patterns.tsv` is the file without wordlist that can be loaded by EDICTOR and inspected, and one file contains the patterns that have been inferred alone, called `D_patterns_Chen.tsv`. In order to inspect the patterns, we recommend to use the EDICTOR tool, which requires the same steps that we already applied when loading our cross-semantic cognates. Once this has been done, press the **ANALYZE** button in the top menu and select **CORRESPONDENCE PATTERNS** in the drop-down menu.



Analyze

Correspondence Patterns

In order to allow for a good display, the doculect names are all abbreviated. Hovering the mouse cursor on an abbreviation will show you the full name.



Doculect

CROSSIDS

Correspondence Patterns

Clicking on a cell in the correspondence pattern panel will allow you to see not only the sound in question, but the full morpheme in which this sound occurs.



Click to Inspect

Click to Expand

4. Validation

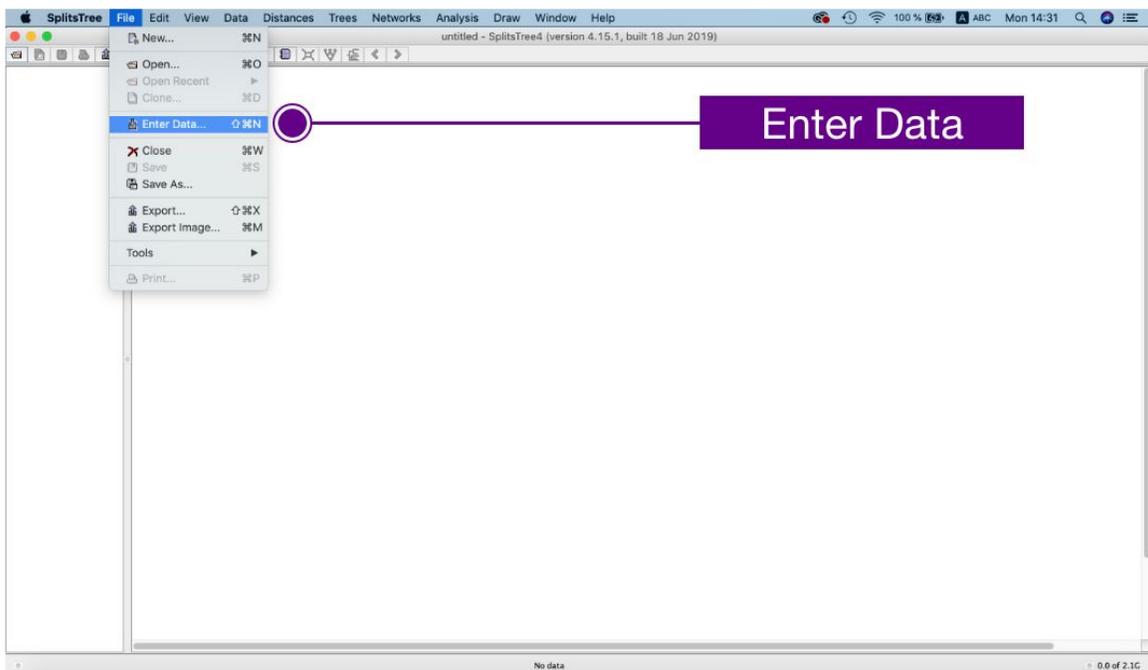
We calculate the shared cognates between language pairs and output the scores in the form of a pairwise distance matrix. The script `6_phylogeny.py` gives two documents, a distance matrix (`A_Chen_distance.dst` or `D_Chen_distance.dst`) and a tree file, based on a Neighbor-Joining analysis (`A_Chen_tree.tre` or `D_Chen_tree.tre`).

There are many ways to work with the distance matrix, here, we give one of the approaches to visualize the matrix as a neighbor-net network with the help of SplitsTree.

To get started, first make sure to install SplitsTree¹⁰ [6] and follow the installation instructions. In order to compute the distance matrix with our code, use the command line (here we compute it for the entire dataset, so we run it with the keyword `all`)

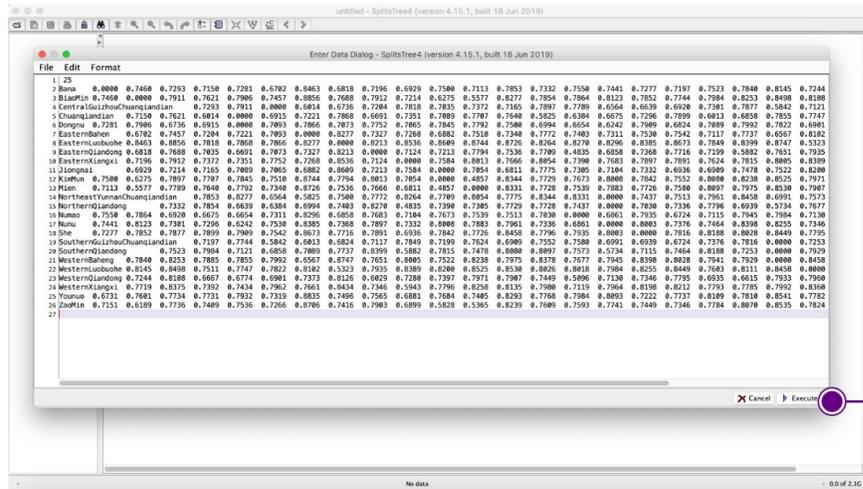
```
$ python 6_phylogeny.py all
```

To generate a Neighbor-Net from the distance matrix, open the file `A_Chen_distance.dst` or `D_Chen_distance.dst` with any plain text editor and start the SplitsTree software. Then click on **File** and **Enter Data**, as shown in the image below.



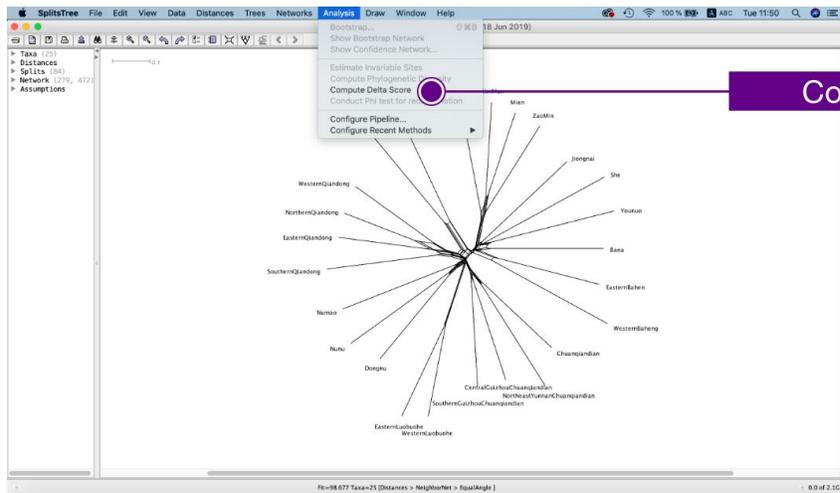
¹⁰ <https://software-ab.informatik.uni-tuebingen.de/download/splitstree4/welcome.html>

Then copy the distance matrix and paste it into the **Enter Data Dialog**, and press **Execute**.



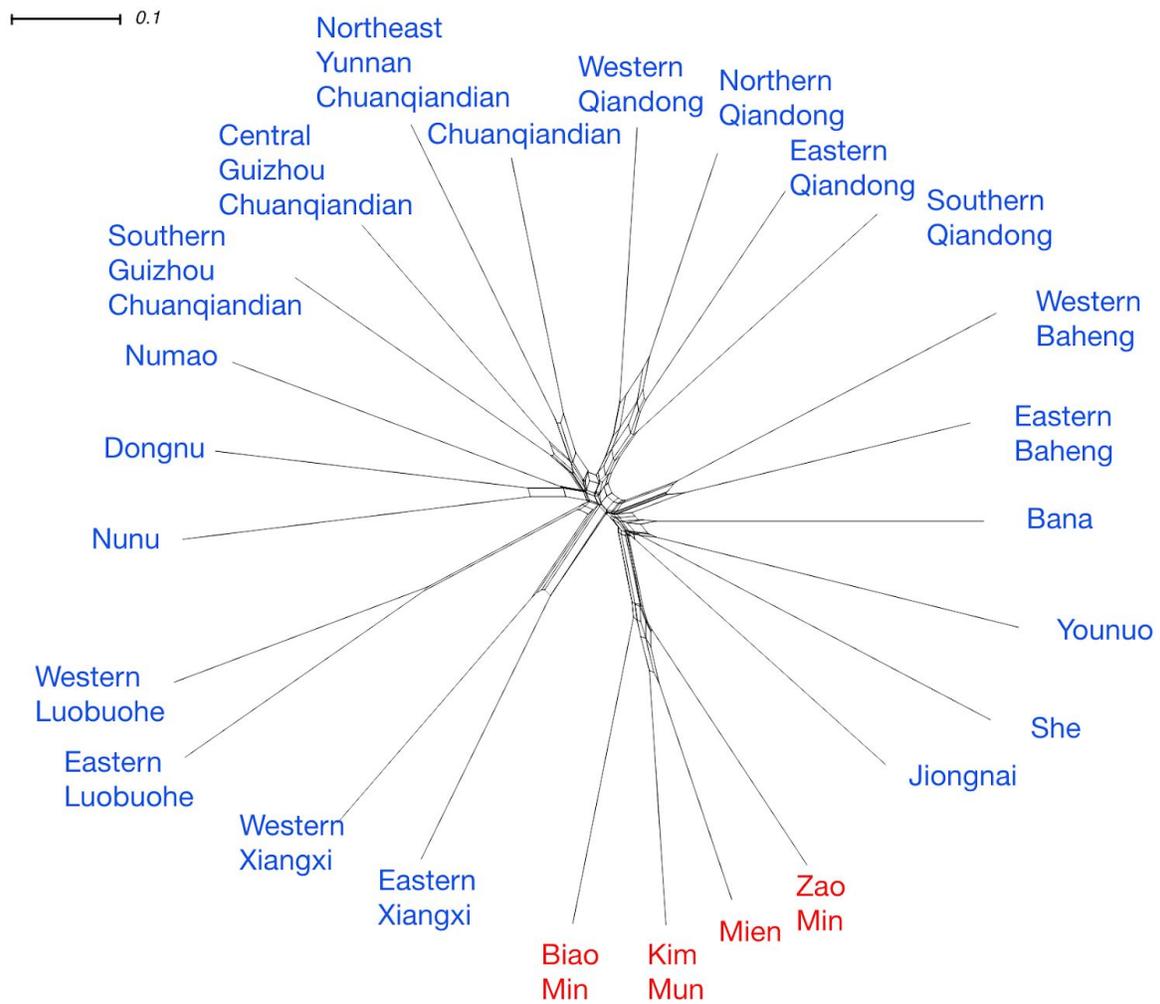
EXECUTE

You can now inspect the network. To analyze the data further, you can compute the delta scores, by pressing **Analysis** and then **Compute Delta Score**, as shown below.



Compute Delta Score

The resulting Neighbor-Net is shown in the following figure. For the purpose of illustration, the Mienic language varieties are colored in red, the Hmongic group is highlighted in blue.



The following table shows the delta scores we computed from the data.

Taxon	Delta score
Bana	0.34706
Biao Min	0.27289
Central Guizhou Chuanqiandian	0.29924
Chuanqiandian	0.29172
Dongnu	0.32416

Eastern Baheng	0.32056
Eastern Luobuohe	0.33529
Eastern Qiangong	0.32083
Eastern Xiangxi	0.33736
Jiongnai	0.32644
Kim Mun	0.26992
Mien	0.25672
Northeast Yunnan Chanqiandian	0.29748
Northern Qiandong	0.28447
Numao	0.34185
Nunu	0.32375
She	0.31671
Southern Guizhou Chuanqiandian	0.34376
Southern Qiandong	0.30988
Western Baheng	0.35259
Western Luobuohe	0.3211
Western Qiandong	0.31137
Western Xiangxi	0.35174
Younuo	0.2996
Zao Min	0.26797

The average delta score is 0.313. As mentioned before, the distances between taxa are calculated via shared cognates. The shorter the distances between two taxa, the higher the similarities between them. If the taxa share cognates not only within their group but also outside their groups, the network finds it challenging to determine the best cluster for them. The larger the reticular structure, or the less tree-like the data is, the higher is the delta score. For one

particular language variety's delta score this means that this specific language contributes to a certain amount of conflict in the data.

5. Conclusion

In this tutorial, we provided details of how to execute our workflow for Computer-Assisted Language comparison, using the scripts we wrote, while at the same time illustrating how the results can be manually inspected and modified. We have not discussed the details of the code we wrote, but we recommend users proficient in Python to have a look.

6. References

1. List J-M, Walworth M, Greenhill SJ, Tresoldi T, Forkel R. Sequence comparison in computational historical linguistics. *Journal of Language Evolution* [Internet]. 2018;3(2):130–44. Available from: <https://academic.oup.com/jole/article/3/2/130/5050100?guestAccessKey=cf8fe64e-3996-4cb1-ba2c-317a7cd81bf4>
2. 陳其光CQ. Miàoyáo yǔwén [Internet]. Běijīng: Zhōngyāng Mínzú Dàxué 中央民族大学 [Central Institute of Minorities]; 2012. Available from: https://en.wiktionary.org/wiki/Appendix:Hmong-Mien_comparative_vocabulary_list
3. List J-M, Lopez P, Bapteste E. Using sequence similarity networks to identify partial cognates in multilingual wordlists. In: *Proceedings of the Association of Computational Linguistics 2016 (Volume 2: Short Papers)* [Internet]. Berlin: Association of Computational Linguistics; 2016. pp. 599–605. Available from: <http://anthology.aclweb.org/P16-2097>
4. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. *Proc Natl Acad Sci USA*. 2008;105(4):1118–23.
5. List J-M. Automatic inference of sound correspondence patterns across multiple languages. *Computational Linguistics* [Internet]. 2019;1(45):137–61. Available from: https://www.mitpressjournals.org/doi/full/10.1162/coli_a_00344
6. Huson DH. SplitsTree: Analyzing and visualizing evolutionary data. *Bioinformatics*. 1998;14(1):68–73.