The `langnames` package[*]

Alejandro García Matarredona

`alejandrogarciaag41@gmail.com`

September 6, 2022

Abstract

The `langnames` package provides a set of macros for formatting names of languages, as well as their identification (in the form of ISO 639-3 codes) and their classification (in the form of its top-level family). The datasets from WALS and Glottolog are included in the package. The package also allows users to rename and add new languages.

Contents

1 Introduction

The typing out of language names in academic papers, especially those in language typology or related fields where many names have to be typed many times, is often inconvenient and inconsistent. This package attempts to be a small help to writers, especially of large projects or of collaborative ones, to have a slightly easier time with names of languages. It does so by defining three main commands: `\lname`, `\liso`, and `\lfam`, which respectively print out the name, name and ISO 639-3 code, and name and family of the specified language. While the package comes with about 7500 pre-defined languages, with code, name, and family, the user may also define new ones through the `\newlang` and `\renewlang` commands. The basic use of all five of these commands is explained below.

---

[*]This document corresponds to `langnames` v2.0, dated 2022/09/05.

2   Usage

2.1   Installation

Download the package from wherever it was found to a place where LATEX may see it, typically in `$TEXMFHOME/tex/latex`. langnames should automatically load the expkv-opt and expkv-def packages.

2.2   Package options

When calling `\usepackage{langnames}`, the user must specify one of three options: `glottolog`, `wals`, or `none`. The first option, `glottolog`, selects the naming conventions from the Glottolog database. The second option, `wals`, predictably selects the naming conventions of the WALS database. The names and the genetic classification differ in some languages, so the user may choose what convention to follow. During the preparation of the dataset, there were instances of languages which appeared in WALS but not in Glottolog, and vice-versa. In such cases, the missing information was added from the other database. For more details on how I built the dataset, one may consult the Python script made for it in the Github repository. The third option, `none`, tells the package not to load either of the datasets, and instead start off from an empty canvas. If one specifies this option, one will have to fill in the details of each language with the macro `\newlang` (see explanation in Section 2.3 below).

2.3   Macros

When referring to a language, the author may use one of three macros to print out different information about it. Languages are identified by their ISO 639-3 code.

`\lname` `\lname{⟨ISO code⟩}`

The simplest macro is `\lname`, which prints out the name of the specified language according to the code provided. The basic syntax is thus`\newlang {⟨ISO code⟩}`. This can be seen in example (1).

(1)   `My native language is \lname{cat}.`
      My native language is Catalan.

`\liso` `\liso{⟨ISO code⟩}`

The `\liso` macro prints out both the name and the ISO 639-3 code of the language specified in the argument (`\liso{⟨ISO code⟩}`). Example (2) shows its behavior.

(2)   `I have recently taken up \liso{brg}.`
      I have recently taken up Baure (ISO 639-3: brg).

\lfam `\lfam{⟨ISO code⟩}`

The `\lfam` command prints the name of the language and its family in parenthesis. Once again, the language is identified by its ISO 639-3 code. Example (3) shows how it works.

(3) `The tone system of \liso{pkt} is fascinating.`
     The tone system of Maleng (ISO 639-3: pkt) is fascinating.

\newlang `\newlang{⟨pseudo code⟩}{⟨name⟩}{⟨family⟩}`

Finally, users may add their own languages via the use of the `\newlang` and `\renewlang` commands. `\newlang` takes three arguments as shown above. Example (4) shows its usage.

(4) `\newlang{boo}{Ameli}{Amelian}`
     `\begin{document}`
     `My new made up language is \lname{boo}.\par\noindent`
     `My new made up language is \liso{boo}.\par\noindent`
     `My new made up language is \lfam{boo}.`
     `\end{document}`

     My new made up language is Ameli.
     My new made up language is Ameli (ISO 639-3: boo).
     My new made up language is Ameli (Amelian).

Note that adding new macros with `\newlang` will not overwrite any from the two other datasets as all three of them have different prefixes. Therefore, if the package was loaded with any option but `none`, none of the `\newlang`s defined will work, because 're meant to be used with one's own dataset.

\renewlang `\renewlang{⟨dataset⟩}{⟨code/pseudo code⟩}{⟨name⟩}{⟨family⟩}`

Unlike `\newlang`, this command actually renews a definition from the specified ⟨*dataset*⟩. This macro therefore has one more argument than `\newlang`, the first argument ⟨*dataset*⟩.

2.3.1 Local changes

The package offers another set of macros to change the language dataset locally, for example if the `wals` option was passed while loading the package and in one section one needs to use the language name from the `glot` set, the following commands may be used. These commands do not take any arguments.

\changetoglottolog
\changetowals
\changetonone

As the names suggest, these will change your dataset for the current local group, i.e., the running environment, or the current pair of {, }, `\begingroup`, `\endgroup` pair, or `\bgroup`, `\egroup` pair.

2.4 A miscellaneous example

The following code:

```
\documentclass{article}
\usepackage[glottolog]{langnames}

\begin{document}
\noindent
My language is \lname{cat}.\par
{%
  \changetonone
  \newlang{cat}{Meow}{Meowian}%
  \noindent
  My language is \lname{cat}.\par
}\noindent
My language is \lname{cat}.\par
\renewlang{glot}{cat}{Meow}{Meowian}\noindent
My language is \lname{cat}.
\end{document}
```

Produces:
My language is Catalan.
My language is Meow.
My language is Catalan.
My language is Meow.

3   Implementation

Language codes, names and families are set with simple `\newcommand`s. These commands have a four part structure as follows:

Internalization: As these commands are for internal use, they should be inaccessible to the users and hence the `@` symbol is used in the command name. The command name starts with the package name, in order to ensure safety. Thus, the first part of macros look like so: `\langnames@`

Name or family: The package presents two different sets of names, namely language names and language family names. Internally they are named `name` and `fams` respectively. Thus, combining the first part with this one one gets `\langnames@names` or `\langname@fams`.

Prefix: As there exist three options, three prefixes need to be defined in order to allow for conditional selection. For this there exist three prefixes: `none`, `wals` and `glot`.

Thus, each language has two internal macros, one defining its name and the other one defining its family, both using the ISO 639-3 code as their key. For more information see `ln_names_*` and `ln_fams_*` files in the package folder.

3.1   Dependencies

The `langnames` package needs to load the `expkv-def` and `expkv-opt` packages for their key-value pair setting functionality.

```
1 \usepackage{expkv-opt,expkv-def}
```

3.2 Option setting

Options are set for what dataset to use. `glottolog` uses Glottolog data; `wals` uses WALS data; `none` selects neither datasets and all languages are defined by the user. See `langnames/langnames.py` in the Github repository to see how I gathered and handled the data.

```
2  \ekvdefinekeys{langnames}{%
3    noval glottolog      = {%
4      \def\langnames@cs@prefix{glot}%
5      \input{ln_names_glot.tex}%
6      \input{ln_fams_glot.tex}%
7    },%
8    noval wals           = {%
9      \def\langnames@cs@prefix{wals}%
10     \input{ln_names_wals.tex}%
11     \input{ln_fams_wals.tex}%
12   },%
13   noval none           = {%
14     \def\langnames@cs@prefix{custom}%
15   }%
16 }
```

This line of code simply tells the package to set the options specified above.

```
17 \ekvoProcessLocalOptions{langnames}
```

3.3 Macro definitions

\lname This macro takes the value specified in its mandatory argument to call its corresponding macro from the `names` set, and prints it. This is achieved through the use of the `\csname` and `\endcsname` macros. The `\unskip` macro is used in all the macro definitions to avoid the adding of an extra space after the macro has been printed.

```
18 \newcommand*{\lname}[1]{%
19   \csname langnames@names@\langnames@cs@prefix @#1\endcsname\unskip
20 }
```

(*End definition for* `\lname`. *This function is documented on page* *2.*)

\liso This macro takes, like `\lname`, the value from the `names` set from the argument input, and prints the name as well as the ISO 639-3 code (which is the argument verbatim) between parenthesis.

```
21 \newcommand*{\liso}[1]{%
22   \csname langnames@names@\langnames@cs@prefix @#1\endcsname{}
23   (ISO 639-3: #1)\unskip
24 }
```

(*End definition for* `\liso`. *This function is documented on page* *2.*)

\lfam This macro, like `\lname` and `\liso`, calls the macro from the `names` set corresponding to the input of the mandatory argument, plus the macro from the `fams` set which gives it its genetic affiliation, which is printed between parenthesis.

```
25 \newcommand*{\lfam}[1]{%
26   \csname langnames@names@\langnames@cs@prefix @#1\endcsname{}
27   (\csname langnames@fams@\langnames@cs@prefix @#1\endcsname)\unskip
28 }
```

\newlang · This macro defines new macros for a language from three mandatory arguments. The first argument of `\newlang{⟨code⟩}` defines the code which serves as identifier (the ISO code in the case of pre-defined key-value pairs). The second argument `{⟨name⟩}` defines the printed name of the language. The third argument `{⟨family⟩}` defines the family to which the language belongs.

```
29 \newcommand*{\newlang}[3]{%
30   \expandafter\def\csname langnames@names@custom@#1\endcsname{#2}%
31   \expandafter\def\csname langnames@fams@custom@#1\endcsname{#3}%
32 }
```

\renewlang · The following code is used to develop the `\renewlang` command.

```
33 \newcommand*{\renewlang}[4]{%
34   \expandafter\def\csname langnames@names@#1@#2\endcsname{#3}%
35   \expandafter\def\csname langnames@fams@#1@#2\endcsname{#4}%
36 }
```

\changetonone
\changetowals
\changetoglottolog
· With the following code these three additional macros are defined which change the macro-set locally.

```
37 \newcommand*{\changetonone}{%
38   \def\langnames@cs@prefix{custom}%
39 }
40 \newcommand*{\changetowals}{%
41   \def\langnames@cs@prefix{wals}%
42 }
43 \newcommand*{\changetoglottolog}{%
44   \def\langnames@cs@prefix{glot}%
45 }
```

This package demands the user to select one package option from the available ones compulsorily. The mechanism of the package might fail if a user does not pass any option. Hence the package checks whether it is passed or not just before the beginning of the document with the following code. If no valid option is passed, an error is issued and the package defaults to the `none` set.

```
46 \def\ssp{\space\space\space\space\space\space}
47 \AddToHook{begindocument/before}{%
48   \ifdefined\langnames@cs@prefix
49   \else
50     \PackageError{langnames}{%
51       You haven't passed any option to 'langnames'. Can't\MessageBreak
52       proceed. Please pass one from the list given below.\MessageBreak
53       ------------------------------------------------\MessageBreak
54       1. glottolog: Glottolog\MessageBreak
55       2. wals:\ssp World Atlas of Languages\MessageBreak
56       3. none:\ssp Your own list.\MessageBreak
57       ------------------------------------------------\MessageBreak
58       Refer to the documentation for more details.\MessageBreak
```

```
59        At the moment I will default to option 'none'%
60      }%
61    \fi
62 }
```