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

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

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` command. The basic use of all four of these commands is explained below.

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,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*⟩`}`

One may also use the `\liso` macro to print out both the name and the ISO 639-3 code of the language specified in the macro in parenthesis, again according to its ISO code (`\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: Arawakan).

\lfam `\lfam{⟨ISO code⟩}`

A third macro for use is the `\lfam` command, which 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{plt} is fascinating.
```
The tone system of Malagasy (ISO 639-3: Austronesian) is fascinating.

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

Finally, users may add their own languages via the use of the `\newlang` command, which 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: Amelian).
My new made up language is Ameli (Amelian).

Be aware that setting a new language overwrites any other language with the same code, as the package only listens to the language that is defined last.

Note that adding new macros won't overwrite any from the two other datasets as all three of them have different prefixes. Also if you have not loaded the package with option `none`; any of your `\newlang`s won't work, because 're supposed to be used with your own dataset.

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

Unlike `\newlang`; this command will actually renew a definition from a particular ⟨*dataset*⟩. This therefore has an extra argument than the former, i.e., the first argument ⟨*dataset*⟩.

2.3.1 Local changes

We additionally have another set of macros to change the language dataset locally, i.e., if you pass the `wals` key while loading the package & in one section you *must* use the language name from the `glot` set, you can use the following commands without 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 `{`, `}`, or a `\begingroup`, `\endgroup` pair, or finally a `\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 & hence we use the `@` symbol in the command name. The command name starts with the package name, to make it more safe. So the first part of our macros look like `\langnames@`

Name or family: We have two different sets of names, namely language names & language family names. Internally they are named `name` & `fams` respectively. Thus combining with this part we get `\langnames@name` or `\langname@fams`.

Prefix: As we have three major modes, we need to define three prefixes in order to allow conditional selection. For this we have three prefixes, i.e., `none`, `wals` & `glot`.

For more information see `ln_langs_*` and `ln_fams_*` files in the package folder. 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.

3.1 Dependencies

The langnames package needs to load the expkv-def,expkv-opt packages for its key-value pair setting functionality.

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

3.2 Option setting

Options are set for what dataset to use. `glottolog` use 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.

```
\ekvdefinekeys{langnames}{%
  noval glottolog       =  {%
    \def\langnames@cs@prefix{glot}%
    \input{ln_langs_glot.tex}%
    \input{ln_fams_glot.tex}%
  },%
  noval wals            =  {%
    \def\langnames@cs@prefix{wals}%
    \input{ln_langs_wals.tex}%
    \input{ln_fams_wals.tex}%
  },%
  noval none            =  {%
    \def\langnames@cs@prefix{custom}%
  }%
}
```

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

```
\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.

```
\newcommand*{\lname}[1]{%
  \csname langnames@name@\langnames@cs@prefix @#1\endcsname\unskip
}
```

(*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.

```
\newcommand*{\liso}[1]{%
  \csname langnames@name@\langnames@cs@prefix @#1\endcsname{}
  (ISO 639-3:
  \csname langnames@fams@\langnames@cs@prefix @#1\endcsname)\unskip
}
```

(*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 the genetic affiliation, which is printed between parenthesis.

```
\newcommand*{\lfam}[1]{%
  \csname langnames@name@\langnames@cs@prefix @#1\endcsname{}
  (\csname langnames@fams@\langnames@cs@prefix @#1\endcsname)\unskip
}
```

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

`\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.

```
\newcommand*{\newlang}[3]{%
  \expandafter\def\csname langnames@name@custom@#1\endcsname{#2}%
  \expandafter\def\csname langnames@fams@custom@#1\endcsname{#3}%
}
```

(*End definition for* `\newlang`. *This function is documented on page 3.*)

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

```
\newcommand*{\renewlang}[4]{%
  \expandafter\def\csname langnames@name@#1@#2\endcsname{#3}%
  \expandafter\def\csname langnames@fams@#1@#2\endcsname{#4}%
}
```

(*End definition for* `\renewlang`. *This function is documented on page 3.*)

`\changetonone` `\changetowals` `\changetoglottolog` With the following code we define these three additional macros to change the macro-set locally.

```
\newcommand*{\changetonone}{%
  \def\langnames@cs@prefix{custom}%
}
\newcommand*{\changetowals}{%
  \def\langnames@cs@prefix{wals}%
}
\newcommand*{\changetoglottolog}{%
  \def\langnames@cs@prefix{glot}%
}
```

(*End definition for* `\changetonone`, `\changetowals`, *and* `\changetoglottolog`. *These functions are documented on page 3.*)

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 doesn't pass any option. Hence we check whether it is passed or not just before the beginning of the document with the following code. If it isn't, we issue an error and default to the `none` set.

```
\def\ssp{\space\space\space\space\space\space}
\AddToHook{begindocument/before}{%
  \ifdefined\langnames@cs@prefix
  \else
    \PackageError{langnames}{%
      You haven't passed any option to 'langnames'. Can't\MessageBreak
      proceed. Please pass one from the list given below.\MessageBreak
      ----------------------------------------------------\MessageBreak
      1. glottolog: Glottolog\MessageBreak
      2. wals:\ssp World Atlas of Languages\MessageBreak
      3. none:\ssp Your own list.\MessageBreak
      ----------------------------------------------------\MessageBreak
      Refer to the documentation for more details.\MessageBreak
      At the moment I will default to option 'none'%
```

```
61      }%
62    \fi
63  }
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.