

Package: hanyupinyin (via r-universe)

May 22, 2026

Title Convert Chinese Characters into Hanyu Pinyin

Version 0.1.3

Description Convert Chinese characters into Hanyu Pinyin (the official romanization system for Standard Chinese) with support for tones, toneless output, initials, URL slugs, and valid R variable names. The package was inspired by the now-orphaned CRAN package 'pinyin' (archived in April 2026 after the maintainer became unreachable). 'hanyupinyin' is a ground-up rewrite using the authoritative Unicode Unihan database, a vectorized engine, and modern R practices. Dictionary data are derived from the Unicode Unihan Database (Unicode Consortium, 2025) <<https://www.unicode.org/reports/tr38/>>.

License MIT + file LICENSE

URL <https://github.com/CuiHR17/hanyupinyin>

BugReports <https://github.com/CuiHR17/hanyupinyin/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5)

Imports stringi

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

Config/pak/sysreqs libicu-dev

Repository <https://cuihr17.r-universe.dev>

Date/Publication 2026-05-20 08:00:24 UTC

RemoteUrl <https://github.com/cuihr17/hanyupinyin>

RemoteRef HEAD

RemoteSha f5711255a733ce769031a025cc7876fe2b02d09c

Contents

add_phrase	2
list_phrases	3
to_pinyin	4
to_pinyin_initials	5
to_pinyin_marks	5
to_pinyin_toneless	6
to_slug	7
to_varname	7
unihan_pinyin	8
Index	9

add_phrase	<i>Add a Custom Polyphone Phrase</i>
------------	--------------------------------------

Description

Allows users to extend the built-in phrase table with their own multi-character phrases and readings. The function automatically detects the input format and stores both a numeric-tone version and a tone-mark version internally, so the phrase works correctly with all settings of the tone argument in `to_pinyin()`.

Usage

```
add_phrase(phrase, reading)
```

Arguments

phrase	A Chinese character string of at least two characters (e.g. "\u884c\u957f").
reading	The corresponding Pinyin reading. Syllables should be separated by spaces (e.g. "hang2 zhang3" or "háng zhǎng"). Underscores (_) and hyphens (-) are also accepted and will be normalised to spaces automatically. Toneless input (e.g. "hang zhang") is allowed but will be treated as-is for both numeric and mark outputs.

Details

The separator used in reading is independent of the sep argument to `to_pinyin()`. The latter controls only the output format.

Value

Invisibly returns NULL.

Examples

```
# Numeric input -- marks are derived automatically
add_phrase("\u884c\u957f", "hang2 zhang3")
to_pinyin("\u94f6\u884c\u884c\u957f", polyphone = TRUE)
to_pinyin("\u94f6\u884c\u884c\u957f", polyphone = TRUE, tone = "marks")

# Tone-mark input -- numeric tones are derived automatically
add_phrase("\u548c\u5e73", "h\u00e9 p\u00edng")
to_pinyin("\u548c\u5e73", polyphone = TRUE, tone = "marks")

# Underscore separators are also accepted
add_phrase("\u6d4b\u8bd5", "ce4_shi4")
to_pinyin("\u6d4b\u8bd5", polyphone = TRUE)
```

list_phrases

List Custom Polyphone Phrases

Description

Returns all user-defined phrases added via `add_phrase()` in the current R session, together with their internally-stored numeric-tone and tone-mark readings.

Usage

```
list_phrases()
```

Value

A data frame with three columns:

phrase The Chinese character phrase.

tone The reading with numeric tones (e.g. "hang2 zhang3").

marks The reading with diacritic tone marks (e.g. "háng zhǎng").

Examples

```
list_phrases()
```

to_pinyin

*Convert Chinese Characters to Hanyu Pinyin***Description**

Converts a character vector of Chinese strings into Pinyin romanization. The function is fully vectorized and uses the Unicode UniHan database (kMandarin) as its authoritative source.

Usage

```
to_pinyin(x, sep = "_", tone = TRUE, polyphone = FALSE, other_replace = NULL)
```

Arguments

x	A character vector.
sep	Separator inserted between syllables in the output . Default is "_". This does not affect how readings are supplied to add_phrase() ; see its documentation for details.
tone	If TRUE (default), returns Pinyin with numeric tones (e.g. qīu1). If FALSE, returns toneless Pinyin (e.g. qiu). If "marks", returns Pinyin with diacritic tone marks (e.g. qīū).
polyphone	If FALSE (default), each character is converted independently using its most common reading from the Unicode UniHan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with add_phrase() .
other_replace	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of the same length as x.

Examples

```
to_pinyin("\u6625\u7720\u4e0d\u89c9\u6653")
to_pinyin("Hello \u4e16\u754c", sep = " ", other_replace = "?")
to_pinyin("\u94f6\u884c\u884c\u957f", polyphone = TRUE)
to_pinyin("\u6625\u7720\u4e0d\u89c9\u6653", tone = "marks")
```

to_pinyin_initials	<i>Extract Pinyin Initials</i>
--------------------	--------------------------------

Description

Returns only the first letter of each syllable.

Usage

```
to_pinyin_initials(x, polyphone = FALSE, other_replace = NULL)
```

Arguments

x	A character vector.
polyphone	If FALSE (default), each character is converted independently using its most common reading from the Unicode Unihan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with add_phrase() .
other_replace	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of the same length as x.

Examples

```
to_pinyin_initials("\u4e2d\u534e\u4eba\u6c11\u5171\u548c\u56fd")
```

to_pinyin_marks	<i>Convert to Pinyin with Tone Marks</i>
-----------------	--

Description

A convenience wrapper around [to_pinyin\(\)](#) with `tone = "marks"`.

Usage

```
to_pinyin_marks(x, sep = "_", polyphone = FALSE, other_replace = NULL)
```

Arguments

x	A character vector.
sep	Separator between syllables. Default is "_".
polyphone	If FALSE (default), each character is converted independently using its most common reading from the Unicode UniHan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with add_phrase() .
other_replace	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of the same length as x.

Examples

```
to_pinyin_marks("\u6625\u7720\u4e0d\u89c9\u6653")
to_pinyin_marks("Hello \u4e16\u754c", sep = " ")
```

to_pinyin_toneless	<i>Convert to Toneless Pinyin</i>
--------------------	-----------------------------------

Description

A convenience wrapper around [to_pinyin\(\)](#) with `tone = FALSE`.

Usage

```
to_pinyin_toneless(x, sep = "_", polyphone = FALSE, other_replace = NULL)
```

Arguments

x	A character vector.
sep	Separator between syllables. Default is "_".
polyphone	If FALSE (default), each character is converted independently using its most common reading from the Unicode UniHan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with add_phrase() .
other_replace	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of the same length as x.

Examples

```
to_pinyin_toneless("\u6625\u7720\u4e0d\u89c9\u6653")
```

to_slug	<i>Create URL-Friendly Slug from Chinese Text</i>
---------	---

Description

Create URL-Friendly Slug from Chinese Text

Usage

```
to_slug(x, polyphone = FALSE, other_replace = NULL)
```

Arguments

x	A character vector.
polyphone	If FALSE (default), each character is converted independently using its most common reading from the Unicode UniHan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with add_phrase() .
other_replace	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of URL-friendly slug strings.

Examples

```
to_slug("2026\u5e74\u62a5\u544a")
```

to_varname	<i>Generate Valid R Variable Names from Chinese Text</i>
------------	--

Description

Useful when cleaning imported data (e.g. from SAS or Excel) where column labels are in Chinese.

Usage

```
to_varname(  
  x,  
  unique = TRUE,  
  abbrev = NULL,  
  polyphone = FALSE,  
  other_replace = NULL  
)
```

Arguments

<code>x</code>	A character vector.
<code>unique</code>	If TRUE (default), appends .1, .2, etc. to duplicates via <code>make.names()</code> .
<code>abbrev</code>	If not NULL, an integer giving the maximum length of each syllable (e.g. <code>abbrev = 4</code> truncates <code>zhong</code> to <code>zhon</code>).
<code>polyphone</code>	If FALSE (default), each character is converted independently using its most common reading from the Unicode Unihan dictionary. If TRUE, a built-in phrase table (50+ common ambiguous words) is used to resolve polyphones via greedy longest-match segmentation. Users can extend the table with <code>add_phrase()</code> .
<code>other_replace</code>	How to handle non-Chinese characters. NULL means leave them as-is. A single character string replaces them.

Value

A character vector of valid R variable names.

Examples

```
to_varname(c("\u59d3\u540d", "\u5e74\u9f84", "\u6027\u522b"))
to_varname("\u4e2d\u534e\u4eba\u6c11\u5171\u548c\u56fd", abbrev = 4)
```

unihan_pinyin

Unihan Pinyin Dictionary

Description

A data frame containing Chinese characters and their Hanyu Pinyin readings extracted from the Unicode Unihan Database (kMandarin field, Version 17.0).

Usage

```
unihan_pinyin
```

Format

A data frame with 44348 rows and 4 variables:

char The Chinese character.

pinyin Pinyin with tone marks (e.g. `qiū`). Multiple readings are space-separated.

pinyin_tone Pinyin with numeric tones (e.g. `qiu1`). Multiple readings are space-separated.

pinyin_toneless Toneless Pinyin (e.g. `qiu`). Multiple readings are space-separated.

Source

Unicode Consortium, Unihan Database, <https://www.unicode.org/reports/tr38/>

Index

* datasets

unihan_pinyin, 8

add_phrase, 2

add_phrase(), 3–8

list_phrases, 3

make.names(), 8

to_pinyin, 4

to_pinyin(), 2, 5, 6

to_pinyin_initials, 5

to_pinyin_marks, 5

to_pinyin_toneless, 6

to_slug, 7

to_varname, 7

unihan_pinyin, 8