Package 'sumer'

January 23, 2026

Type Package

Title Sumerian Cuneiform Text Analysis

Version 1.0.0

Description Provides functions for converting transliterated Sumerian texts to sign names and cuneiform characters, creating and querying dictionaries, and analyzing the structure of Sumerian words. Includes a built-in dictionary and supports both forward lookup (Sumerian to English) and reverse lookup (English to Sumerian).

License GPL-3

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stringr, officer, xml2, dplyr, cli

RoxygenNote 7.3.3

Maintainer Robin Wellmann <ro.wellmann@gmail.com>

NeedsCompilation no

Author Robin Wellmann [aut, cre]

Repository CRAN

Date/Publication 2026-01-23 14:20:02 UTC

Contents

`sumer-package` *Tools for Working with Sumerian Cuneiform Texts*

Description

A toolkit for analyzing and translating Sumerian cuneiform texts. The package provides functions for converting sign names to cuneiform characters, creating and querying dictionaries, and analyzing the structure of Sumerian words.

Getting Started

Load the package and explore the built-in dictionary:

```
library(sumer)

# Load the built-in dictionary
dic <- read_dictionary()

# Look up a Sumerian word
look_up("d-suen", dic)

# Search for a term in translations
look_up("water", dic, "en")
```

Cuneiform Conversion and Analysis

Functions for converting sign names to cuneiform and analyzing sign structure:

`as.cuneiform` Converts Sumerian text to cuneiform characters.

```
as.cuneiform("lugal")
as.cuneiform("AN.EN.ZU")
```

`as.sign_name` Converts Sumerian text to sign names.

```
as.sign_name("lugal")
```

`split_sumerian` Splits compound sign names into individual components.

```
split_sumerian("AN.EN.ZU")
```

`info` Displays detailed information about a Sumerian text.

```
info("jic-tukul")
```

Dictionary Creation

Functions for creating dictionaries from annotated translation files:

`read_translated_text` Reads annotated translation files (.docx or .txt) and extracts sign names, grammatical types, and meanings.

```
filename    <- system.file("extdata", "text_with_translations.txt", package = "sumer")
translations <- read_translated_text(filename)
```

`convert_to_dictionary` Converts translation data into dictionary format, adding cuneiform representations and phonetic readings.

```
dictionary <- convert_to_dictionary(translations)
```

`make_dictionary` Convenience function that combines reading and conversion in one step.

```
dictionary <- make_dictionary(filename)
```

Dictionary Input/Output

Functions for saving and loading dictionaries:

`save_dictionary` Saves a dictionary to a text file with metadata header.

```
save_dictionary(dic, "my_dictionary.txt",
                author = "John Doe",
                version = "1.0")
```

`read_dictionary` Loads a dictionary from file. Without arguments, loads the built-in dictionary.

```
dic <- read_dictionary()
dic <- read_dictionary("my_dictionary.txt")
```

Dictionary Lookup

`look_up` Interactive dictionary search. Supports both forward lookup (Sumerian to English) and reverse lookup (English to Sumerian).

```
# Forward lookup: Sumerian sign name
look_up("AN", dic)
look_up("AN.EN.ZU", dic)

# Reverse lookup: search in translations
look_up("king", dic, "en")
look_up("Gilgamesh", dic, "en")
```

`skeleton` Creates a structured template (skeleton) for translating Sumerian text.

```
skeleton("e-ta-na an-ce3 ba-ed3-de3")
```

Typical Workflows

Workflow A: Analyze and translate a text:

```
library(sumer)

# Convert sign name to cuneiform
as.cuneiform("d-en-líl")

# Load dictionary
dic <- read_dictionary()

# Look up the meaning
look_up("d-en-líl", dic)

# Get information about individual signs
info("d-en-líl")
info("lil")
info("lil2")
```

Workflow B: Create your own dictionary:

```
library(sumer)

# Read annotated translations from file
filename    <- system.file("extdata", "text_with_translations.txt", package = "sumer")
translations <- read_translated_text(filename)

# Convert to dictionary format
dictionary <- convert_to_dictionary(translations)

# Save for later use
save_dictionary(dictionary, "my_dictionary.txt",
                author  = "My Name",
                year    = "2025",
                version = "1.0")

# Load and use
my_dic <- read_dictionary("my_dictionary.txt")
look_up("ki", my_dic)
```

Author(s)

Maintainer: [Robin Wellmann] <ro.wellmann@gmail.com>

See Also

Core functions: info, read_dictionary, look_up

Dictionary creation: read_translated_text, convert_to_dictionary, make_dictionary, save_dictionary skeleton

Analysis tools: split_sumerian, as.cuneiform, as.sign_name

---

as.cuneiform *Convert Transliterated Sumerian Text to Cuneiform*

Description

Converts transliterated Sumerian text to Unicode cuneiform characters. This is a generic function with a method for character vectors.

Usage

```
as.cuneiform(x, ...)

## Default S3 method:
as.cuneiform(x, ...)

## S3 method for class 'character'
as.cuneiform(x, mapping = NULL, ...)

## S3 method for class 'cuneiform'
print(x, ...)
```

Arguments

x	For as.cuneiform: An object to be converted to cuneiform. Currently, only character vectors are supported. For print.cuneiform: an object of class "cuneiform".
mapping	A data frame containing the sign mapping table with columns syllables, name, and cuneiform. If NULL (the default), the package's internal mapping file 'etcsl_mapping.txt' is loaded. Only used by the character method.
...	Additional arguments passed to methods.

Details

The function processes each element of the input character vector by:

1. Calling info to look up sign information for each transliterated sign.
2. Extracting the Unicode cuneiform symbols for each sign.
3. Reconstructing the cuneiform text using the original separators, but removing hyphens and periods which are only used in transliteration to indicate sign boundaries.

The default method throws an error for unsupported input types.

Value

`as.cuneiform` returns a character vector of class `cuneiform` with the cuneiform representation of each input element.

`print.cuneiform` displays a character vector of class `cuneiform`.

Note

The cuneiform output requires a font that supports the Unicode Cuneiform block (U+12000 to U+12500) to display correctly.

See Also

`info` for retrieving detailed sign information, `split_sumerian` for splitting Sumerian text into signs, `as.sign_name` for converting transliterated Sumerian text intos sign names

Examples

```
# Convert transliterated text to cuneiform
as.cuneiform(c("na-an-jic li-ic ma","en tarah-an-na-ke4"))

# Load transliterated text from a file
file <- system.file("extdata", "transliterated-text.txt", package = "sumer")
x <- readLines(file)
cat(x, sep="\n")

# Convert transliterated text to cuneiform
as.cuneiform(x)

# Using a custom mapping table
path <- system.file("extdata", "etcsl_mapping.txt", package = "sumer")
my_mapping <- read.csv2(path, sep=";", na.strings="")
as.cuneiform("lugal", mapping = my_mapping)

# Error for unsupported types

as.cuneiform(123)
# Error in as.cuneiform.default(123) : Cannot coerce to cuneiform
```

---

as.sign_name *Convert Transliterated Sumerian Text to Sign Names*

Description

Converts transliterated Sumerian text to canonical sign names in uppercase notation. This is a generic function with a method for character vectors.

Usage

```
as.sign_name(x, ...)

## Default S3 method:
as.sign_name(x, ...)

## S3 method for class 'character'
as.sign_name(x, mapping = NULL, ...)

## S3 method for class 'sign_name'
print(x, ...)
```

Arguments

x	For as.sign_name: An object to be converted to sign names. Currently, only character vectors are supported. For print.sign_name: An object of class "sign_name".
mapping	A data frame containing the sign mapping table with columns syllables, name, and cuneiform. If NULL (the default), the package's internal mapping file 'etcsl_mapping.txt' is loaded. Only used by the character method.
...	Additional arguments passed to methods.

Details

The function processes each element of the input character vector by:

1. Calling info to look up sign information for each transliterated sign.
2. Extracting the canonical sign names for each sign.
3. Reconstructing the text using the original separators, but replacing hyphens with periods to follow standard sign name notation.

The default method throws an error for unsupported input types.

Value

as.sign_name returns a character vector of class c("sign_name", "character") with the sign name representation of each input element.

print.sign_name displays a character vector of class "sign_name".

See Also

as.cuneiform for converting to cuneiform characters, info for retrieving detailed sign information, split_sumerian for splitting Sumerian text into signs

Examples

```
# Convert transliterated text to sign names
as.sign_name(c("lugal-e", "an-ki"))

# Load transliterated text from a file
file <- system.file("extdata", "transliterated-text.txt", package = "sumer")
x <- readLines(file)
cat(x, sep="\n")

# Convert transliterated text to sign names
as.sign_name(x)

# Using a custom mapping table
path <- system.file("extdata", "etcsl_mapping.txt", package = "sumer")
my_mapping <- read.csv2(path, sep=";", na.strings="")
as.sign_name("lugal", mapping = my_mapping)

# Error for unsupported types

as.sign_name(123)
# Error in as.sign_name.default(123) : Cannot coerce to cuneiform names
```

---

convert_to_dictionary *Convert Translation Data to a Sumerian Dictionary*

Description

Converts a data frame of Sumerian translations into a structured dictionary format, adding cuneiform representations and phonetic readings for each sign.

Usage

```
convert_to_dictionary(df, mapping = NULL)
```

Arguments

`df`	A data frame with columns `sign_name`, `type`, and `meaning`, typically produced by `read_translated_text`.
`mapping`	A data frame containing sign-to-reading mappings with columns `name`, `cuneiform` and `syllables`. If `NULL` (default), the package's built-in mapping file `etcsl_mapping.txt` is used.

Details

Processing Steps:

1. Aggregates translations and counts occurrences of each unique combination in `df`
2. Looks up phonetic readings and cuneiform signs for each sign component

3. Combines cuneiform, reading, and translation rows into a single data frame
4. Sorts the result by sign name and row type

Reading Format: Phonetic readings are formatted as follows:

- Multiple possible readings are enclosed in braces: `{a, dur5, duru5}`
- For compound signs, readings of individual components are joined with hyphens
- If a sign has more than three possible readings in a compound, only the first three are shown followed by `...`
- Unknown readings are marked with ?

Value

A data frame with the following columns:

sign_name The normalized Sumerian text (e.g., `"A"`, `"AN"`, `"A2.TAB"`)

row_type Type of entry: `"cunei."` (cuneiform character), `"reading"` (phonetic readings), or `"trans."` (translation)

count Number of occurrences for translations; `NA` for cuneiform and reading entries

type Grammatical type (e.g., `"S"`, `"V"`, `"A"`) for translations; empty string for other row types

meaning The cuneiform character(s), phonetic reading(s), or translated meaning depending on `row_type`

The data frame is sorted by `sign_name`, `row_type`, and descending `count`.

See Also

`read_translated_text` for reading translation files, `make_dictionary` for creating a complete dictionary with cuneiform representations and readings in a single step.

Examples

```
# Read translations from a single text document
filename     <- system.file("extdata", "text_with_translations.txt", package = "sumer")
translations <- read_translated_text(filename)

# View the structure
head(translations)

#Make some custom unifications (here: removing the word "the")
translations$meaning <- gsub("\\bthe\\b", "", translations$meaning, ignore.case = TRUE)
translations$meaning <- trimws(gsub("\\s+", " ", translations$meaning))

# View the structure
head(translations)

#Convert the result into a dictionary
dictionary   <- convert_to_dictionary(translations)

# View the structure
```

```
head(dictionary)

# View entries for a specific sign
dictionary[dictionary$sign_name == "EN", ]

# With custom mapping
path  <- system.file("extdata", "etcsl_mapping.txt", package = "sumer")
mapping <- read.csv2(path, sep=";", na.strings="")
translations <- read_translated_text(filename, mapping = mapping)
dictionary <- convert_to_dictionary(translations, mapping = mapping)
head(dictionary)
```

`info` *Retrieve Information About Sumerian Signs*

Description

Analyzes a transliterated Sumerian text string and retrieves detailed information about each sign, including syllabic readings, sign names, cuneiform symbols, and alternative readings.

The function `info` computes the result and returns an object of class `"info"`. The `print` method displays a summary of different text representations in the console.

Usage

```
info(x, mapping = NULL)

## S3 method for class 'info'
print(x, flatten = FALSE, ...)
```

Arguments

`x`	For `info`: a character string of length 1 containing transliterated Sumerian text. For `print.info`: an object of class `"info"`.
`mapping`	A data frame containing the sign mapping table with columns `syllables`, `name`, and `cuneiform`. If `NULL` (the default), the package's internal mapping file '`etcsl_mapping.txt`' is loaded.
`flatten`	Logical. If `TRUE`, grammar indicators in the text are removed (such as parentheses, brackets, braces, and operators). If `FALSE` (the default), the original separators are preserved.
`...`	Additional arguments passed to the print method (currently unused).

Details

The function `info` performs the following steps:

1. Splits the input string into signs and separators using `split_sumerian`.
2. Standardizes the signs.

3. Looks up each sign in the mapping table based on its type:
   - Type 1 (lowercase): Searches for a matching syllable reading.
   - Type 2 (uppercase): Searches for a matching sign name.
   - Type 3 (cuneiform): Searches for a matching cuneiform character.
4. Returns a data frame with the results, along with the separators stored as an attribute.

The mapping table must contain the following columns:

syllables Comma-separated list of possible syllabic readings for the sign. The first reading is used as the default.

name The canonical sign name in uppercase.

cuneiform The Unicode cuneiform character.

The `print` method displays each sign with its name and alternative readings, followed by three text representations: syllables, sign names, and cuneiform text.

Value

`info` returns a data frame of class `c("info", "data.frame")` with one row per sign and the following columns:

`reading`	The syllabic reading of the sign. For lowercase input, this is the standardized input; for other types, this is the default syllable from the mapping.
`sign`	The Unicode cuneiform character corresponding to the sign.
`name`	The canonical sign name in uppercase.
`alternatives`	A comma-separated string of all possible syllabic readings for the sign.

The data frame has an attribute `"separators"` containing the separator characters between signs.

`print.info` prints the following to the console and returns `x` invisibly:

Sign table Each sign with its cuneiform symbol, name, and alternative readings.

syllables The text with syllabic readings, using hyphens as separators within words.

sign names The text with sign names, using periods as separators within words.

cuneiform text The text rendered in Unicode cuneiform characters, with hyphens and periods removed.

Note

If no custom mapping is provided, the function loads the internal mapping file included with the **sumer** package.

See Also

`split_sumerian` for splitting Sumerian text into signs,

Examples

```
library(stringr)

# Basic usage - compute and print
info("lugal-e")

# Store the result for further processing
result <- info("an-ki")
result

# Access the underlying data frame
result$sign
result$name

# Print with and without flattened separators
result <- info("(an)na")
print(result)
print(result, flatten = TRUE)

# Using a custom mapping table
path <- system.file("extdata", "etcsl_mapping.txt", package = "sumer")
my_mapping <- read.csv2(path, sep=";", na.strings="")
info("an-ki", mapping = my_mapping)
```

---

look_up *Look Up Sumerian Signs or Search for Translations*

Description

Searches a Sumerian dictionary either by sign name (forward lookup) or by translation text (reverse lookup).

The function `look_up` computes the search results and returns an object of class `"look_up"`. The `print` method displays formatted results with cuneiform representations, grammatical types, and translation counts.

Usage

```
look_up(x, dic, lang = "sumer", width = 70)

## S3 method for class 'look_up'
print(x, ...)
```

Arguments

`x`	For `look_up`: A character string specifying the search term. Can be either: • A Sumerian sign name (e.g., `"AN"`, `"AN.EN.ZU"`) • A cuneiform character string • A word or phrase to search in translations (e.g., `"Gilgamesh"`, `"heaven"`)

	For `print.look_up`: An object of class `"look_up"` as returned by `look_up`.
`dic`	A dictionary data frame, typically created by `make_dictionary` or loaded with `read_dictionary`. Must contain columns `sign_name`, `row_type`, `count`, `type`, and `meaning`.
`lang`	Character string specifying whether `x` is a Sumerian expression (`"sumer"`) or an English expression (`"en"`).
`width`	Integer specifying the text width for line wrapping. Default is 70.
`...`	Additional arguments passed to the print method (currently unused).

Details

Search Modes: The function operates in two modes depending on the input:

Forward Lookup (Sumerian input detected):

1. Converts the sign name to cuneiform
2. Retrieves all translations for the exact sign combination
3. Retrieves translations for all individual signs and substrings

Reverse Lookup (non-Sumerian input):

1. Searches for the term in all translation meanings
2. Retrieves matching entries with sign names and cuneiform

Output Format: The print method displays results with:

- Sign names with cuneiform representations
- Occurrence counts in brackets (e.g., `[29]`)
- Grammatical type abbreviations (e.g., `S`, `V`)
- Translation meanings with automatic line wrapping
- Search term highlighting in blue for reverse lookups (only for ANSI-compatible terminals)

Value

`look_up` returns an object of class `"look_up"`, which is a list containing:

`search`	The original search term.
`lang`	The language setting used for the search.
`width`	The text width for formatting.
`cuneiform`	The cuneiform representation (only for Sumerian searches).
`sign_name`	The canonical sign name (only for Sumerian searches).
`translations`	A data frame with translations for the exact sign combination (only for Sumerian searches).
`substrings`	A named list of data frames with translations for individual signs and substrings (only for Sumerian searches).
`matches`	A data frame with matching entries (only for non-Sumerian searches).

`print.look_up` prints formatted dictionary entries to the console and returns `x` invisibly.

See Also

read_dictionary for loading dictionaries, make_dictionary for creating dictionaries, as.cuneiform for cuneiform conversion.

Examples

```
# Load dictionary
dic <- read_dictionary()

# Forward lookup: search by phonetic spelling
look_up("d-suen", dic)

# Forward lookup: search by Sumerian sign name
look_up("AN", dic)
look_up("AN.EN.ZU", dic)

# Forward lookup: search by cuneiform character string
AN.NA <- paste0(intToUtf8(0x1202D), intToUtf8(0x1223E))
AN.NA
look_up(AN.NA, dic)

# Reverse lookup: search in translations
look_up("Gilgamesh", dic, "en")

# Adjust output width for narrow terminals
look_up("water", dic, "en", width = 50)

# Store results for later use
result <- look_up("lugal", dic)
result$cuneiform
result$translations

# Print stored results
print(result)
```

---

make_dictionary *Create a Sumerian Dictionary from Annotated Text Files*

Description

Parses Word documents (.docx) or plain text files containing annotated Sumerian translations and creates a structured dictionary data frame. The function extracts sign names, their cuneiform representations, possible readings, and translations with grammatical types.

Usage

```
make_dictionary(file, mapping = NULL)
```

Arguments

file	A character vector of file paths to .docx or text files. Files must contain translation lines that are formatted as described below.
mapping	A data frame containing sign-to-reading mappings with columns name, cuneiform and syllables. If NULL (default), the package's built-in mapping file etcsl_mapping.txt is used.

Details

Input Format: The input files must contain lines starting with | in the following format:

|sign_name: TYPE: meaning

or

|equation for sign_name: TYPE: meaning

For example:

```
|a2-tab: S: the double amount of work performance
|me=ME: S: divine force
|AN: S: god of heaven
|na=NA: Sx->A: whose existence is bound to S
```

Lines not starting with | are ignored. Only the first entry in an equation of sign names is used for the dictionary. The following notation is suggested for grammatical types:

- S for substantives and noun phrases, (e.g., "the old man in the temple")
- V for verbs and decorated verbs (e.g., "to go", "to bring the delivery into the temple")
- A for adjectives, attributes and subordinate clauses that further define the subject (e.g., "who/which is weak", "whose resource for sustaining life is grain")
- Sx->A for a symbol that transforms the preceding noun phrase into an attribute (e.g., "whose resource for sustaining life is S"). Other transformations are denoted accordingly.
- N for numbers,
- D for everything else.

Processing Steps:

1. Extracts text from .docx files or reads plain text
2. Filters lines starting with |
3. Normalizes sign names and looks up possible readings from the mapping table
4. Aggregates translations and counts occurrences

Output Structure: For each unique sign, the output contains:

- One cunei. row with the cuneiform character(s)
- One reading row with possible phonetic readings
- One or more trans. rows with translations, sorted by frequency

Value

A data frame with the following columns:

sign_name The normalized Sumerian sign name (e.g., "A", "AN", "ME")

line_type Type of entry: `"cunei."` (cuneiform), `"reading"` (phonetic readings), or `"trans."` (translation)

count Number of occurrences for translations; `NA` for cuneiform and reading entries

type Grammatical type (e.g., "S", "V", "Sx->A") for translations; empty for other line types

meaning The cuneiform character(s), reading(s), or translated meaning depending on line_type

See Also

`as.cuneiform`, `split_sumerian`

Examples

```
# Create a dictionary from a single text document
filename  <- system.file("extdata", "text_with_translations.txt", package = "sumer")
dict <- make_dictionary(filename)

# Use the dictionary
look_up("an", dict)
```

`read_dictionary` *Read a Sumerian Dictionary from File*

Description

Reads a Sumerian dictionary from a semicolon-separated text file, optionally displaying the metadata header with author, version, and update information.

Usage

```
read_dictionary(file = NULL, verbose = TRUE)
```

Arguments

`file`	A character string specifying the path to the dictionary file. If `NULL` (default), the package's built-in dictionary `sumer-dictionary.txt` is loaded.
`verbose`	Logical. If `TRUE` (default), the metadata header (author, year, version, URL) is printed to the console.

Details

File Format: The function expects a semicolon-separated file with a metadata header. Lines starting with # are treated as comments. The expected format is:

```
###-----------------------------------------------------------------
###                Sumerian Dictionary
###
### Author:  Robin Wellmann
### Year:    2026
### Version: 0.5
### Watch for Updates:
###   https://founder-hypothesis.com/en/sumerian-mythology/downloads/
###-----------------------------------------------------------------
sign_name;row_type;count;type;meaning
A;cunei.;;;<here would be the cuneiform sign for A>
A;reading;;;{a, dur5, duru5}
A;trans.;3;S;water
```

Encoding: The file is read with UTF-8 encoding to properly handle cuneiform characters.

Value

A data frame with the following columns:

sign_name The Sumerian sign name (e.g., `"A"`, `"AN"`, `"ME"`)

row_type Type of entry: `"cunei."` (cuneiform character), `"reading"` (phonetic readings), or `"trans."` (translation)

count Number of occurrences for translations; `NA` for cuneiform and reading entries

type Grammatical type (e.g., `"S"`, `"V"`) for translations; empty string for other row types

meaning The cuneiform character(s), phonetic reading(s), or translated meaning depending on `row_type`

See Also

`save_dictionary` for saving dictionaries to file, `make_dictionary` and `convert_to_dictionary` for creating dictionaries.

Examples

```
# Load the built-in dictionary
dic <- read_dictionary()

# Load a custom dictionary
filename <- system.file("extdata", "sumer-dictionary.txt", package = "sumer")
dic <- read_dictionary(filename)

# Look up an entry
look_up("d-suen", dic)
```

`read_translated_text` *Read Annotated Sumerian Translations from Text Files*

Description

Reads Word documents (.docx) or plain text files containing annotated Sumerian translations and extracts sign names, grammatical types, and meanings into a structured data frame.

Usage

```
read_translated_text(file, mapping=NULL)
```

Arguments

`file`	A character vector of file paths to .docx or text files. Files must contain translation lines that are formatted as described below.
`mapping`	A data frame containing sign-to-reading mappings with columns `name`, `cuneiform` and `syllables`. If `NULL` (default), the package's built-in mapping file `etcsl_mapping.txt` is used.

Details

Input Format: The input files must contain lines starting with `|` in the following format:

`|sign_name: TYPE: meaning`

or

`|equation for sign_name: TYPE: meaning`

For example:

```
|a2-tab: S: the double amount of work performance
|me=ME: S: divine force
|AN: S: god of heaven
|na=NA: Sx->A: whose existence is bound to S
```

Lines not starting with `|` are ignored. Only the first entry in an equation of sign names is extracted. The following notation is suggested for grammatical types:

- `S` for substantives and noun phrases, (e.g., "the old man in the temple")
- `V` for verbs and decorated verbs (e.g., "to go", "to bring the delivery into the temple")
- `A` for adjectives, attributes and subordinate clauses that further define the subject (e.g., "who/which is weak", "whose resource for sustaining life is grain")
- `Sx->A` for a symbol that transforms the preceding noun phrase into an attribute (e.g., "whose resource for sustaining life is `S`"). Other transformations are denoted accordingly.
- `N` for numbers,
- `D` for everything else.

Processing Steps:

1. Reads text from .docx files or plain text files

2. Filters lines starting with |
3. Parses each line into sign name, type, and meaning components
4. Normalizes transliterated text by removing separators and looking up the sign names from the mapping
5. Cleans meaning field by removing content after ; or | delimiters
6. Issues a warning for entries with missing type annotations
7. Excludes empty sign names from the result

Value

A data frame with the following columns:

sign_name The normalized sign name with components separated by hyphens (e.g., "A", "AN", "X-NA")

type Grammatical type (e.g., "S", "V", "A", "Sx->A")

meaning The translated meaning of the sign

Note

If any translations have missing type annotations, the function prints a warning message listing the affected entries.

See Also

convert_to_dictionary for converting the result into a dictionary, make_dictionary for creating a complete dictionary with cuneiform representations and readings in a single step.

Examples

```
# Read translations from a single text document
filename     <- system.file("extdata", "text_with_translations.txt", package = "sumer")
translations <- read_translated_text(filename)

# View the structure
head(translations)

# Filter by grammatical type
nouns <- translations[translations$type == "S", ]
nouns

#Make some custom unifications (here: removing the word "the")
translations$meaning <- gsub("\\bthe\\b", "", translations$meaning, ignore.case = TRUE)
translations$meaning <- trimws(gsub("\\s+", " ", translations$meaning))

# View the structure
head(translations)

#Convert the result into a dictionary
dictionary   <- convert_to_dictionary(translations)
```

```
# View the structure
head(dictionary)
```

`save_dictionary` *Save a Sumerian Dictionary to File*

Description

Saves a Sumerian dictionary data frame to a semicolon-separated text file with a metadata header containing author, year, version, and URL information.

Usage

```
save_dictionary(dic, file, author = "", year = "", version = "", url = "")
```

Arguments

`dic`	A dictionary data frame, typically created by `make_dictionary` or `convert_to_dictionary`. Must contain columns `sign_name`, `row_type`, `count`, `type`, and `meaning`.
`file`	A character string specifying the output file path.
`author`	A character string with the author name(s) for the metadata header.
`year`	A character string with the year of creation for the metadata header.
`version`	A character string with the version number for the metadata header.
`url`	A character string with a URL where updates can be found.

Details

Output Format: The output file consists of two parts:

1. A metadata header with lines starting with ###, containing author, year, version, and URL information
2. The dictionary data in semicolon-separated format with columns: `sign_name`, `row_type`, `count`, `type`, `meaning`

Example output:

```
###------------------------------------------------------------
###                Sumerian Dictionary
###
### Author:  Robin Wellmann
### Year:    2026
### Version: 1.0
### Watch for Updates: https://founder-hypothesis.com/sumer/
###------------------------------------------------------------
sign_name;row_type;count;type;meaning
A;cunei.;;;<cuneiform sign for A>
A;reading;;;{a, dur5, duru5}
A;trans.;3;S;water
```

Value

No return value. The function is called for its side effect of writing the dictionary to a file.

See Also

`make_dictionary` and `convert_to_dictionary` for creating dictionaries, `read_dictionary` for reading saved dictionaries.

Examples

```
# Create and save a dictionary

filename  <- system.file("extdata", "text_with_translations.txt", package = "sumer")
dictionary <- make_dictionary(filename)

save_dictionary(
  dic     = dictionary,
  file    = "sumerian_dictionary.txt",
  author  = "John Doe",
  year    = "2024",
  version = "1.0",
  url     = "https://example.com/dictionary"
)
```

---

skeleton *Create a Translation Template for Sumerian Text*

Description

Creates a structured template (skeleton) for translating Sumerian text. The template displays each word and syllable with its sign name and cuneiform representation, providing a framework for adding translations.

The function `skeleton` computes the template and returns an object of class `"skeleton"`. The `print` method displays the template in the console.

Usage

```
skeleton(x, mapping = NULL)

## S3 method for class 'skeleton'
print(x, ...)
```

Arguments

x	For skeleton: A character string of length 1 containing transliterated Sumerian text. Words are separated by spaces, syllables within words by hyphens or other separators. For print.skeleton: An object of class "skeleton" as returned by skeleton.
mapping	A data frame containing the sign mapping table with columns syllables, name, and cuneiform. If NULL (the default), the package's internal mapping file 'etcsl_mapping.txt' is loaded.
...	Additional arguments passed to the print method (currently unused).

Details

The function generates a hierarchical template with different levels of detail depending on the input type:

Multiple words The template includes a header line with the original text, followed by entries for each word, its syllables (indented with one tab), and sub-signs for multi-sign syllables (indented with two tabs).

Single word (multiple syllables) The word equation serves as the header, followed by syllable entries (one tab) and sub-sign entries (two tabs). No redundant header line is generated.

Single syllable Only the syllable equation is shown (no indentation), with sub-sign entries indented by one tab if applicable.

Each line in the template follows the format

`|[tabs]reading=SIGN.NAME=cuneiform::`

The template should be filled in as follows:

- Between the two colons: the grammatical type of the expression (e.g., S for noun phrases, V for verbs, etc.). See make_dictionary for details.
- After the second colon: the translation

For example, a filled-in line might look like:

`|an=AN=<cuneiform sign for AN>:S: god of heaven`

Redundant lines are automatically omitted: if a word consists of only one syllable, no separate syllable line is generated.

This function is intended to be used together with look_up for translating Sumerian texts: first create a template with skeleton, then use look_up to find the meanings of words and signs, and fill in the template accordingly.

The template format is designed to be saved as a text file (.txt) or Word document (.docx), filled in manually, and can then be used as input for make_dictionary to create a custom dictionary.

Value

skeleton returns a character vector of class c("skeleton", "character") containing the template lines.

print.skeleton prints the template to the console and returns x invisibly.

See Also

look_up for looking up translations of Sumerian signs and words, make_dictionary for creating a dictionary from filled-in templates, info for retrieving detailed sign information

Examples

```
# Create a template for a multi-word phrase
skeleton("e-ta-na an-ce3 ba-ed3-de3")

# Create a template for a single word
skeleton("lugal-e")

# Create a template for a single syllable
skeleton("an")

# Store the template for further use
tmpl <- skeleton("lu2 du")
tmpl

# Typical workflow: create template, then look up meanings
dic <- read_dictionary()
tmpl <- skeleton("lugal kur-ra-ke4")
print(tmpl)
look_up("lugal", dic)
look_up("kur", dic)
```

---

split_sumerian *Split a String into Sumerian Signs and Separators*

Description

Splits a transliterated Sumerian text string into its constituent signs and the separators between them. The function recognizes three types of Sumerian sign representations: lowercase transliterations, uppercase sign names, and Unicode cuneiform characters.

Usage

```
split_sumerian(x)
```

Arguments

x	A character string containing transliterated Sumerian text.

Details

The function identifies Sumerian signs based on three patterns:

1. **Lowercase transliterations** (type 1): Sequences of lowercase letters (a-z) including special characters (ĝ, š, ...) and accented vowels (á, é, í, ú, à, è, ì, ù), optionally followed by a numeric index.

2. **Uppercase sign names** (type 2): Sequences starting with an uppercase letter, optionally followed by additional uppercase letters, digits, or the characters +, /, and ×.
3. **Cuneiform characters** (type 3): Unicode characters in the Cuneiform block (U+12000 to U+12500).

The function returns the signs and separators in a format that allows exact reconstruction of the original string using `paste0(c("", signs), separators, collapse = "")`.

Value

A list with three components:

`signs`	A character vector containing the extracted Sumerian signs.
`separators`	A character vector of length `length(signs) + 1` containing the separators. The first element contains any text before the first sign, subsequent elements contain text between consecutive signs, and the last element contains any text after the final sign. Empty strings indicate no separator at that position.
`types`	An integer vector of the same length as `signs` indicating the type of each sign: `1` for lowercase transliterations, `2` for uppercase sign names, and `3` for cuneiform characters.

Examples

```
# Example 1
x <- "en-tarah-an-na-ke4"

result <- split_sumerian(x)

result

# Example 2

x <- "en-DARA3.AN.na-ke4"

result <- split_sumerian(x)

result

# Reconstruct the original string
paste0(c("", result$signs), result$separators, collapse = "")
```

Index