

# Package: hunspell (via r-universe)

August 21, 2024

**Type** Package

**Title** High-Performance Stemmer, Tokenizer, and Spell Checker

**Version** 3.0.4

**Depends** R (>= 3.0.2)

**Encoding** UTF-8

**Description** Low level spell checker and morphological analyzer based on the famous 'hunspell' library <<https://hunspell.github.io>>. The package can analyze or check individual words as well as parse text, latex, html or xml documents. For a more user-friendly interface use the 'spelling' package which builds on this package to automate checking of files, documentation and vignettes in all common formats.

**License** GPL-2 | LGPL-2.1 | MPL-1.1

**URL** <https://docs.ropensci.org/hunspell/>  
<https://ropensci.r-universe.dev/hunspell>

**BugReports** <https://github.com/ropensci/hunspell/issues>

**Imports** Rcpp, digest

**LinkingTo** Rcpp (>= 0.12.12)

**Suggests** spelling, testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**Language** en-US

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/hunspell>

**RemoteRef** master

**RemoteSha** 0e177e5c897d50233063a36b5ab05d6f10ab7ef6

## Contents

hunspell . . . . . 2

hunspell

*Hunspell Spell Checking and Morphological Analysis***Description**

The [hunspell](#) function is a high-level wrapper for finding spelling errors within a text document. It takes a character vector with text (plain, latex, man, html or xml format), parses out the words and returns a list with incorrect words for each line. It effectively combines [hunspell\\_parse](#) with [hunspell\\_check](#) in a single step. Other functions in the package operate on individual words, see details.

**Usage**

```

hunspell(
  text,
  format = c("text", "man", "latex", "html", "xml"),
  dict = dictionary("en_US"),
  ignore = en_stats
)

hunspell_parse(
  text,
  format = c("text", "man", "latex", "html", "xml"),
  dict = dictionary("en_US")
)

hunspell_check(words, dict = dictionary("en_US"))

hunspell_suggest(words, dict = dictionary("en_US"))

hunspell_analyze(words, dict = dictionary("en_US"))

hunspell_stem(words, dict = dictionary("en_US"))

hunspell_info(dict = dictionary("en_US"))

dictionary(lang = "en_US", affix = NULL, add_words = NULL, cache = TRUE)

list_dictionaries()

```

**Arguments**

text	character vector with arbitrary input text
format	input format; supported parsers are text, latex, man, xml and html.
dict	a dictionary object or string which can be passed to <a href="#">dictionary</a> .

ignore	character vector with additional approved words added to the dictionary
words	character vector with individual words to spell check
lang	dictionary file or language, see details
affix	file path to corresponding affix file. If NULL it is assumed to be the same path as dict with extension .aff.
add_words	a character vector of additional words to add to the dictionary
cache	speed up loading of dictionaries by caching

## Details

Hunspell uses a special dictionary format that defines which stems and affixes are valid in a given language. The `hunspell_analyze` function shows how a word breaks down into a valid stem plus affix. The `hunspell_stem` function is similar but only returns valid stems for a given word. Stemming can be used to summarize text (e.g in a wordcloud). The `hunspell_check` function takes a vector of individual words and tests each one for correctness. Finally `hunspell_suggest` is used to suggest correct alternatives for each (incorrect) input word.

Because spell checking is usually done on a document, the package includes some parsers to extract words from various common formats. With `hunspell_parse` we can parse plain-text, latex and man format. R also has a few built-in parsers such as `RdTextFilter` and `SweaveTeXFilter`, see also `?aspell`.

The package searches for dictionaries in the working directory as well as in the standard system locations. `list_dictionaries` provides a list of all dictionaries it can find. Additional search paths can be specified by setting the DICPATH environment variable. A US English dictionary (en\_US) is included with the package; other dictionaries need to be installed by the system. Most operating systems already include compatible dictionaries with names such as `hunspell-en-gb` or `myspell-en-gb`.

To manually install dictionaries, copy the corresponding .aff and .dic file to `~/Library/Spelling` or a custom directory specified in DICPATH. Alternatively you can pass the entire path to the .dic file as the dict parameter. Some popular sources of dictionaries are `SCOWL`, `OpenOffice`, `debian`, `github/titoBouzout` or `github/woorm`.

Note that hunspell uses `iconv` to convert input text to the encoding used by the dictionary. This will fail if text contains characters which are unsupported by that particular encoding. For this reason UTF-8 dictionaries are preferable over legacy 8-bit dictionaries.

## Examples

```
# Check individual words
words <- c("beer", "wiskey", "wine")
correct <- hunspell_check(words)
print(correct)

# Find suggestions for incorrect words
hunspell_suggest(words[!correct])

# Extract incorrect from a piece of text
bad <- hunspell("spell checkers are not neccessairy for langauge ninja's")
print(bad[[1]])
```

```
hunspell_suggest(bad[[1]])

# Stemming
words <- c("love", "loving", "lovingly", "loved", "lover", "lovely", "love")
hunspell_stem(words)
hunspell_analyze(words)

# Check an entire latex document
tmpfile <- file.path(tempdir(), "1406.4806v1.tar.gz")
download.file("https://arxiv.org/e-print/1406.4806v1", tmpfile, mode = "wb")
untar(tmpfile, exdir = tempdir())
text <- readLines(file.path(tempdir(), "content.tex"), warn = FALSE)
bad_words <- hunspell(text, format = "latex")
sort(unique(unlist(bad_words)))

# Summarize text by stems (e.g. for wordcloud)
allwords <- hunspell_parse(text, format = "latex")
stems <- unlist(hunspell_stem(unlist(allwords)))
words <- head(sort(table(stems), decreasing = TRUE), 200)
```

# Index

`?aspell`, [3](#)

`dicpath(hunspell)`, [2](#)

`dictionary`, [2](#)

`dictionary(hunspell)`, [2](#)

`en_stats(hunspell)`, [2](#)

`hunspell`, [2](#), [2](#)

`hunspell_analyze`, [3](#)

`hunspell_analyze(hunspell)`, [2](#)

`hunspell_check`, [2](#), [3](#)

`hunspell_check(hunspell)`, [2](#)

`hunspell_find(hunspell)`, [2](#)

`hunspell_info(hunspell)`, [2](#)

`hunspell_parse`, [2](#), [3](#)

`hunspell_parse(hunspell)`, [2](#)

`hunspell_stem`, [3](#)

`hunspell_stem(hunspell)`, [2](#)

`hunspell_suggest`, [3](#)

`hunspell_suggest(hunspell)`, [2](#)

`iconv`, [3](#)

`list_dictionaries`, [3](#)

`list_dictionaries(hunspell)`, [2](#)

`RdTextFilter`, [3](#)

`SweaveTeXFilter`, [3](#)