# Package 'TeXCheckR'

January 20, 2025

**Type** Package

**Title** Parses LaTeX Documents for Errors

**Date** 2024-02-04

**Version** 0.8.1

**URL** https://github.com/HughParsonage/TeXCheckR

**BugReports** https://github.com/HughParsonage/TeXCheckR/issues

**Description** Checks LaTeX documents and .bib files for typing errors, such as spelling errors, incorrect quotation marks. Also provides useful functions for parsing and linting bibliography files.

**License** GPL-2

**Depends** R (>= 3.3.0)

**Imports** clisymbols, crayon, data.table (>= 1.9.0), fastmatch, hunspell (>= 2.5), hutils (>= 0.8.0), magrittr, rstudioapi, stats, tools, zoo

**LazyData** TRUE

**ByteCompile** true

**RoxygenNote** 7.2.0

**Encoding** UTF-8

**Suggests** devtools, readr, rlang, stringi, testthat (>= 2.0.0), tinytex

**NeedsCompilation** no

**Author** Hugh Parsonage [aut, cre]

**Maintainer** Hugh Parsonage <hugh.parsonage@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-02-04 03:20:02 UTC

## Contents

TeXCheckR-package . . . . . 3
any_bib_duplicates . . . . . 3
argument_parsing . . . . . 4

bib_parser . . . 5
braces_closes_at . . . 6
check_biber . . . 7
check_consecutive_words . . . 7
check_dashes . . . 8
check_escapes . . . 9
check_footnote_typography . . . 9
check_labels . . . 10
check_literal_citations . . . 11
check_literal_xrefs . . . 11
check_quote_marks . . . 12
check_spelling . . . 12
check_xrefs . . . 15
commands_used . . . 15
correctly_spelled_words . . . 16
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE . . . 16
extract_LaTeX_argument . . . 17
extract_mandatory_LaTeX_argument . . . 17
extract_optional_LaTeX_argument . . . 18
extract_validate_abbreviations . . . 19
figs_tbls_unrefd . . . 19
inputs_of . . . 20
isR_line_in_knitr . . . 20
lint_bib . . . 21
locate_mandatory_LaTeX_argument . . . 21
minimal_bib . . . 22
parse_tex . . . 22
position_of_string . . . 23
read_tex_document . . . 24
report_error . . . 24
rm_editorial_square_brackets . . . 25
separate_sentences . . . 26
split_report . . . 26
strip_comments . . . 27
tex_group_by_char . . . 28
validate_bibliography . . . 28
valid_English_contractions . . . 29
veto_sic . . . 30
weld_bmillion . . . 30
wrongly_spelled_words . . . 31

**Index** **32**

`TeXCheckR-package` *TeXCheckR*

## Description

Checks LaTeX documents and `.bib` files for typing errors, such as spelling errors, incorrect quotation marks. Also provides useful functions for parsing and linting bibliography files.

`any_bib_duplicates` *Are any bib entries duplicated?*

## Description

Are any bib entries duplicated?

## Usage

```
any_bib_duplicates(bib.files, .report_error, rstudio = FALSE)
```

## Arguments

| | |
|---|---|
| `bib.files` | Files to check for duplicates |
| `.report_error` | How errors should be logged. |
| `rstudio` | Use the RStudio API? |

## Details

This function is very fastidious about the format of `bib.files`. Run `lint_bib` (noting that this will overwrite your bibliography) if it complains.

This function finds exact duplicates in the author title date/year and volume fields. Note that it is not possible in general to detect actual duplicates; you will still need to inspect the printed bibliography.

## Value

Called for its side-effect. If duplicates are detected, the first six are printed as a `data.table`; otherwise, `NULL`, invisibly.

argument_parsing *Replace nth arguments*

## Description

Replace nth arguments

## Usage

```
replace_nth_LaTeX_argument(
  tex_lines,
  command_name,
  n = 1L,
  replacement = "correct",
  optional = FALSE,
  warn = TRUE,
  .dummy_replacement = "Qq"
)

nth_arg_positions(
  tex_lines,
  command_name,
  n = 1L,
  optional = FALSE,
  star = TRUE,
  data.tables = TRUE,
  allow_stringi = TRUE
)
```

## Arguments

| | |
|---|---|
| tex_lines | A character vector of a LaTeX file (as read in from readLines or readr::read_lines). |
| command_name | The command name, or the pattern of the command, without the initial backslash. |
| n | Which argument of the command. |
| replacement | What to replace the nth argument with. |
| optional | If FALSE, the default, the nth mandatory argument is extracted. If TRUE, the nth *optional* argument is extracted. |
| warn | If the nth argument is not present, emit a warning? Set to FALSE for n-ary commands. |
| .dummy_replacement | An intermediate replacement value. This value cannot be present in tex_lines. |
| star | Assume the starred version of the command. That is, assume that the contents of the argument lies on a single line. |
| data.tables | Should each element of the list be a data.table? Set to FALSE for performance. |
| allow_stringi | (logical, default: TRUE) If FALSE, non-stringi functions are allowed. |

### Details

nth_arg_positions reports the starts and stops of the command for every line. This includes the braces (in order to accommodate instances where the argument is empty).

If the line is empty or does not contain the command the values of starts and stops are NA_integer_.

### Examples

```
nth_arg_positions("This is a \\textbf{strong} statement.", "textbf")
replace_nth_LaTeX_argument("This is a \\textbf{strong} statement.", "textbf")
```

---

## bib_parser *Functions for parsing .bib files*

---

### Description

Functions for parsing .bib files

### Usage

```
fread_bib(
  file.bib,
  check.dup.keys = TRUE,
  strip.braces = TRUE,
  check.unescaped.percent = TRUE,
  .bib_expected = TRUE,
  halt = TRUE,
  rstudio = FALSE,
  .report_error
)

bib2DT(file.bib, to_sort = FALSE)

reorder_bib(file.bib, outfile.bib = file.bib)
```

### Arguments

| | |
|---|---|
| file.bib | .bib file. |
| check.dup.keys | If TRUE, the default, return error if any bib keys are duplicates. |
| strip.braces | If TRUE, the default, braces in fields are removed. |
| check.unescaped.percent | If TRUE, the default, fields with unescaped percent signs are an error. (Unescaped percent signs in URLs are permitted.) Set to FALSE to skip this check. |
| .bib_expected | (logical, default: TRUE) Should file.bib be expected to have file extension .bib? If expectation violated, a warning is emitted. |

| | |
|---|---|
| halt | Whether to halt on error. If NULL, the default, the value getOption("TeXCheckR.halt_on_error") is used. Otherwise, TRUE or FALSE to halt regardless of the value of the option. |
| rstudio | (logical, default: FALSE) If TRUE, pop the RStudio session to the location in file.bib of the first error. |
| .report_error | A function like report2console to handle errors. |
| to_sort | Include only author, title, year, and date. |
| outfile.bib | File to write the reordered bib to. Defaults to file.bib. |

## Details

bib2DT returns a data.table of the entries in file.bib. The function reorder_bib rewrites file.bib, to put it in surname, year, title, line number order.

---

# braces_closes_at *Brace closes at*

---

## Description

Where do braces close?

## Usage

```
braces_closes_at(tex_line, position_of_opening_brace)
```

## Arguments

| | |
|---|---|
| tex_line | A single line. |
| position_of_opening_brace | An integer giving the position of the opening brace in question. |

## Value

The positions of the closing brace matching the opening braces at position_of_opening_brace.

## check_biber *Check biber*

### Description

Check biber

### Usage

```
check_biber(path = ".", rstudio = FALSE)
```

### Arguments

| | |
|---|---|
| path | The path containing the blg file, following successful compilation. |
| rstudio | Use the RStudio API? |

## check_consecutive_words *Check consecutive typeset words*

### Description

Check consecutive typeset words

### Usage

```
check_consecutive_words(
  path = ".",
  latex_file = NULL,
  md5sum.ok = NULL,
  outfile = NULL,
  outfile.append = FALSE
)
```

### Arguments

| | |
|---|---|
| path | Path containing the LaTeX file. |
| latex_file | The LaTeX file (without path) whose output will be checked. |
| md5sum.ok | The output of md5sum of an acceptable LaTeX file. Since some repeated words will be spurious, you can use the md5sum of the output of this function. |
| outfile | A file to which the output can be saved. If NULL, the default, the output is printed to the console (and not saved). |
| outfile.append | (logical, default: FALSE). Append or overwrite outfile if specified? If FALSE, the default, and file exists, outfile will be overwritten. |

## Value

NULL if the LaTeX document does not create a PDF with lines repeated. An error if words are repeated on consecutive lines, together with cat() output of the offending lines. The output is presented in 'stanzas':

```
'<Repeated word>'
        <Context>
```

for example a document that results in the following lines, notably the repetition of *household*, the output would be:

```
'household'
       affordable. This `mortgage burden' is often defined as the proportion of
        household income spent on repaying a mortgage. Depending on the
        household income measure used, the mortgage burden on a newly
        purchased first home, assuming a person borrows 80 per cent of the
        value of the home, is currently lower than much of the period between
```

Lastly the error message contains the md5sum of the file is returned in the error message, so it can be supplied to md5sum.ok.

---

check_dashes *Check dashes entered as hyphens*

---

## Description

Check dashes entered as hyphens

## Usage

```
check_dashes(
  filename,
  .report_error,
  dash.consistency = c("en-dash", "em-dash"),
  protases_ok = TRUE,
  rstudio = TRUE
)
```

## Arguments

| | |
|---|---|
| filename | A tex or Rnw file. |
| .report_error | How errors should be reported. |
| dash.consistency | Character vector permitted dash types. |
| protases_ok | (logical, default: TRUE) Should em-dashes be permitted when they form a protasis in a list? \item when there is an emdash---always. |
| rstudio | (logical, default: TRUE) Use the RStudio API? |

### Value

File stops and cat()s on any line where a hyphen is surrounded by a space. Excludes dashes in knitr chunks and LaTeX math mode \(...\) but not in TeX math mode $...$.

---

## check_escapes *Check escapes*

---

### Description

Checks file for unescaped dollar signs. With these present, there is a risk of constructions like We gave $10 to a million people at a cost of $10~million dollars., which is valid syntax, but incorrectly formatted. Accordingly, math-mode must be more assertively requested using \(..\).

### Usage

```
check_escapes(filename, .report_error)
```

### Arguments

| | |
|---|---|
| filename | File in which to report the error |
| .report_error | How the errors should be reported. |

### Value

An error if unescaped dollar signs are present in filename. Otherwise, NULL invisibly.

---

## check_footnote_typography *Check footnote typography*

---

### Description

Check footnote typography

### Usage

```
check_footnote_typography(
  filename,
  ignore.lines = NULL,
  .report_error,
  rstudio = FALSE
)
```

### Arguments

| | |
|---|---|
| filename | A LaTeX file. |
| ignore.lines | Lines to ignore (for example, those using the word 'footnote'). |
| .report_error | A function to provide context to any errors. |
| rstudio | (logical, default: FALSE) Should the RStudio API be used? |

### Details

See https://github.com/grattan/grattex/blob/master/doc/grattexDocumentation.pdf for full set of error conditions.

### Value

Called for its side-effect.

### Examples

```
## Not run:
  tex_file <- tempfile(fileext = ".tex")
  cat("Footnote not ending with full stop.\\footnote{No sentence}", file = tex_file)
  check_footnote_typography(tex_file)

## End(Not run)
```

---

## check_labels *Check labels*

---

### Description

Check labels

### Usage

```
check_labels(filename, .report_error, check.chaprefs = TRUE)
```

### Arguments

| | |
|---|---|
| filename | The LaTeX source file to check. |
| .report_error | The function to provide context to the error. |
| check.chaprefs | (logical, default: TRUE) If TRUE, require all cross-references to use \Chapref. |

### Details

Checks each label has a prefix and the prefix is one of the following: fig:, tbl:, box:, chap:, sec:, eq:, subsec:, subsubsec:, para: paragraph:. Checks also that chapter labels are marked with chap:. (N.B. although each label must have a prefix, it must not necessarily the *right* prefix; for example, a table caption may have prefix tbl:.)

### Value

NULL, invisibly if labels check out. An error otherwise.

---

## check_literal_citations *Check that citations are all using cites*

---

### Description

Check that citations are all using cites

### Usage

```
check_literal_citations(filename, .report_error)
```

### Arguments

| | |
|---|---|
| filename | TeX document |
| .report_error | Function to report errors |

---

## check_literal_xrefs *Check for hard-coded cross-references*

---

### Description

Check for hard-coded cross-references

### Usage

```
check_literal_xrefs(filename, .report_error)
```

### Arguments

| | |
|---|---|
| filename | The TeX file to check |
| .report_error | How errors should be reported. |

### Value

An error, or if none found, NULL invisibly.

## check_quote_marks *Check quote marks in TeX*

### Description

Checks whether a closing quote has been used at the start of a word.

### Usage

```
check_quote_marks(filename, .report_error, rstudio = FALSE)
```

### Arguments

| | |
|---|---|
| filename | LaTeX filename. |
| .report_error | A function determining how errors will be reported. |
| rstudio | Use the rstudioapi package to jump to the location of the first error. |

### Examples

```
## Not run:
  tex_file <- tempfile(fileext = ".tex")
  cat("This is the wrong 'quote' mark.", file = tex_file)
  check_quote_marks(tex_file)
  file.remove(tex_file)

## End(Not run)
```

## check_spelling *Spell checking*

### Description

Spell checking

### Usage

```
check_spelling(
  filename,
  tex_root = dirname(filename),
  pre_release = TRUE,
  ignore.lines = NULL,
  known.correct = NULL,
  known.correct.fixed = NULL,
  known.wrong = NULL,
  ignore_spelling_in = NULL,
```

```
  ignore_spelling_in_nth = NULL,
  bib_files,
  check_etcs = TRUE,
  dict_lang = "en_GB",
  rstudio = FALSE,
  .report_error
)
```

## Arguments

| | |
|---|---|
| filename | Path to a LaTeX file to check. |
| tex_root | The root path of the filename. Provide this if you are checking an \input file that has a different root directory to its parent. |
| pre_release | Should the document be assumed to be final? Setting to FALSE permits the use of ignore_spelling_in and permits add_to_dictionary to be present outside the document preamble. |
| ignore.lines | Integer vector of lines to ignore (due to possibly spurious errors). |
| known.correct | Character vector of patterns known to be correct (which will never be raised by this function). |
| known.correct.fixed | Character vector of words known to be correct (which will never be raised by this function). |
| known.wrong | Character vector of patterns known to be wrong. |
| ignore_spelling_in | Command whose first mandatory argument will be ignored. |
| ignore_spelling_in_nth | Named list of arguments to ignore; names are the commands to be ignored, values are the nth argument to be ignored. |
| bib_files | Bibliography files (containing possible clues to misspellings). If supplied, and this function would otherwise throw an error, the .bib files are read and any author names that match the misspelled words are added to the dictionary. |
| check_etcs | If TRUE, stop if any variations of etc, ie, and eg are present. (If they are typed literally, they may be formatted inconsistently. Using a macro ensures they appear consistently.) |
| dict_lang | Passed to hunspell::dictionary. |
| rstudio | Use the RStudio API? |
| .report_error | A function to provide context to any errors. If missing, defaults to report2console. |

## Details

Extends and enhances hunspell:

- You can add directives in the document itself. To add a word foobaz to the dictionary (so its presence does not throw an error), write % add_to_dictionary: foobaz on a single line. The advantage of this method is that you can collaborate on the document without having to keep track of which spelling errors are genuine.

- The directive `% ignore_spelling_in: mycmd` which will ignore the spelling of words within the first argument of `\mycmd`.
- `ignore_spelling_in_file: <file.tex>` will skip the check of `<file.tex>` if it is `input` or `include` in `filename`, as well as any files within it. Should appear as it is within `input` but with the file extension
- Only the root document need be supplied; any files that are fed via `\input` or `\include` are checked (recursively).
- A historical advantages was that the contents of certain commands were not checked, the spelling of which need not be checked as they are not printed, `viz.` citation and cross-reference commands, and certain optional arguments. Most of these are now parsed correctly by `hunspell`, though some still need to be supplied (including, naturally, user-supplied macros).
- Abbreviations and initialisms which are validly introduced will not throw errors. See `extract_valid_abbrevations`.
- Words preceded by '[sic]' will not throw errors.

The package comes with a suite of `correctly_spelled_words` that were not present in `hunspell`'s dictionary.

This function should be quite fast, but slower than `hunspell::hunspell` (which it invokes). I aim for less than 500 ms on a real-world report of around 100 pages. The function is slower when it needs to consult `bib_files`, though I recommend adding authors, titles, etc. to the dictionary explicitly, or using `citeauthor` and friends.

This function is forked from `https://github.com/hughparsonage/grattanReporter` to parse reports of the Grattan Institute, Melbourne for errors. See `https://github.com/grattan/grattex/blob/master/doc/grattexDocumentation.pdf` for the full spec. Some checks that package performs have been omitted in this package.

## Value

Called primarily for its side-effect. If the spell check fails, the line at which the first error was detected, with an error message. If the check succeeds, `NULL` invisibly.

## Examples

```
## Not run:
url_bib <-
paste0("https://raw.githubusercontent.com/HughParsonage/",
       "grattex/e6cab97145d38890e44e83d122e995e3b8936fc6/",
       "Report.tex")
check_spelling(url_bib)

## End(Not run)
```

## check_xrefs *Check cross-references*

### Description

Check cross-references that are repetitive or (in the case of `cleveref` and `varioref`) incorrect case.

### Usage

```
check_xrefs(filename, permitted.case = c(NA, "upper", "lower"), .report_error)
```

### Arguments

| | |
|---|---|
| `filename` | A LaTeX file |
| `permitted.case` | One of `NA`, `"upper"`, `"lower"`. If `NA`, the default, both `\Cref` and `\cref` are permitted, but not in the same document. If `upper`, only `\Cref` is permitted; if `lower`, only `\cref`. If `NULL`, the case is not checked at all. |
| `.report_error` | The function to provide context to the error. |

## commands_used *List all unique commands in a document*

### Description

List all unique commands in a document

### Usage

```
commands_used(tex_lines)
```

### Arguments

| | |
|---|---|
| `tex_lines` | A LaTeX document as read from `readr::read_lines` or `readLines`. |

### Value

A character vector of unique commands used in `tex_lines`.

### Examples

```
commands_used(c("A \\abc{d}", "\\def{x}"))
```

---

`correctly_spelled_words`
*List of correctly spelled words*

---

## Description

List of correctly spelled words

## Usage

```
correctly_spelled_words
```

## Format

A character vector of words as perl-regex patterns to skip during the spell check.

---

`CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE`
*List of correctly spelled, case-sensitive words*

---

## Description

List of correctly spelled, case-sensitive words

## Usage

```
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE
```

## Format

A character vector of words as perl-regex case-sensitive patterns to skip during the spell check.

## extract_LaTeX_argument

*Extract LaTeX command argument*

### Description

This is a simple wrapper around `extract_mandatory_LaTeX_argument` and `extract_optional_LaTeX_argument`.

### Usage

```
extract_LaTeX_argument(tex_lines, command_name, n = 1L, optional = FALSE)
```

### Arguments

| | |
|---|---|
| `tex_lines` | LaTeX text. |
| `command_name` | Name of command without backslash `\textbf` corresponds to `command_name = "textbf"`. |
| `n` | Which argument to extract, if exists. |
| `optional` | Extract the optional argument, rather than the mandatory arguments. |

## extract_mandatory_LaTeX_argument

*Extract mandatory argument II*

### Description

Extract mandatory argument II

### Usage

```
extract_mandatory_LaTeX_argument(
  tex_lines,
  command_name,
  n = 1L,
  by.line = FALSE,
  parsed_doc = NULL
)
```

### Arguments

| | |
|---|---|
| tex_lines | A character vector of lines as read from a LaTeX document. |
| command_name | The command name (no backslash or opening brace). |
| n | Which integer to |
| by.line | If FALSE, the default, each row of the data.table returned has the entire contents of the argument in extract column. If TRUE, the contents is split as it is in the document; arguments over multiple lines in the document are split over multiple rows in the data.table returned. |
| parsed_doc | A parsed document (from parse_tex). parse_tex. Use this argument if the cost of running parse_tex is expensive (such as repeatedly over the same document). |

---

## extract_optional_LaTeX_argument

*Extract optional argument*

---

### Description

Extract optional argument

### Usage

```
extract_optional_LaTeX_argument(
  tex_lines,
  command_name,
  n = 1L,
  by.line = FALSE
)
```

### Arguments

| | |
|---|---|
| tex_lines | A character vector reading from a LaTeX document. |
| command_name | Name of command (without backslash) |
| n | Which optional argument to extract. |
| by.line | Should the output be one row per command (FALSE, the default), with extracts concatenated via paste0(..., collapse = "") or one row per line per command? |

## extract_validate_abbreviations *Extract valid abbreviations and initialisms*

### Description

Extracts abbreviations which are preceded by the full text (*e.g.* 'The Quebec Xylophone Enterprise Foundation (QXEF)').

### Usage

```
extract_validate_abbreviations(lines)
```

### Arguments

| | |
|---|---|
| lines | Lines to extract |

### Details

Only 'valid' abbreviations are extracted, viz. those abbreviations of the form (ABC) where the first letters of the preceding words (excluding some common words like of, and, etc.) are 'a', 'b', 'c'.

### Value

Character vector of abbreviations of the form (ABC)

## figs_tbls_unrefd *Return unreferenced figures or tables in document*

### Description

Useful for checking whether all the figures and tables in a document have been referenced in the main text. You may exclude figures and tables from the check by using the directive % may_be_left_unreferenced: in the preamble before the label that is to be excluded.

### Usage

```
figs_tbls_unrefd(filename, .report_error, check.labels = TRUE)
```

### Arguments

| | |
|---|---|
| filename | A LaTeX file. |
| .report_error | A function to provide context to any errors. |
| check.labels | if TRUE, the default, run check_labels on filename to ensure the figure and table labels in filename are in the expected form or style. Set to FALSE for possibly faster runs but the risk of spurious results. |

### Value

The labels of any figure or table left unreferenced in `filename` (including inputs).

---

`inputs_of` *Inputs to files nested within LaTeX document*

---

### Description

Inputs to files nested within LaTeX document

### Usage

```
inputs_of(filename, exclude.preamble = TRUE, append.tex = TRUE)
```

### Arguments

| | |
|---|---|
| `filename` | The file whose `\inputs` are to be extracted. |
| `exclude.preamble` | (logical) If `TRUE`, the default, only `\inputs` and `\includes` within the `document` environment are returned. |
| `append.tex` | Should the result include the file extension `.tex`? By default, `TRUE`. Setting to `FALSE` may be useful when the file is not a `.tex` file. |

### Value

A character vector of file paths relative to `filename` that are used as `\inputs` or `\includes` within `filename`. If no such files are present within `filename`, `NULL` is returned.

---

`isR_line_in_knitr` *Is a line in knitr R or not?*

---

### Description

Is a line in knitr R or not?

### Usage

```
isR_line_in_knitr(lines)
```

### Arguments

| | |
|---|---|
| `lines` | Lines to check, as in the result of `readLines`. Not a filename. |

### Value

`TRUE` if in knitr chunk (including boundaries). `FALSE` otherwise.

## lint_bib *Tidy bibliography so equals signs align*

### Description

Tidy bibliography so equals signs align

### Usage

```
lint_bib(bib_file, outfile = bib_file, leading_spaces = 2L)
```

### Arguments

| | |
|---|---|
| bib_file | The bib file to tidy. |
| outfile | Optionally, the tidied bib file to write to. |
| leading_spaces | The number of spaces before each field within an entry. |

### Details

Aligns the equals signs in `bib_file` and ensures all fields have a trailing comma.

## locate_mandatory_LaTeX_argument *Locate contents of LaTeX commands*

### Description

Provides the locations of LaTeX commands with mandatory arguments.

### Usage

```
locate_mandatory_LaTeX_argument(
  tex_lines,
  command_name,
  n = 1L,
  parsed_doc = NULL
)
```

### Arguments

| | |
|---|---|
| tex_lines | A character vector of a LaTeX document, – for example as obtained from `readLines("mydoc.tex")`. |
| command_name | The command (without backslash) whose arguments' locations are desired. |
| n | Integer vector: which argument(s) to locate. If `n = NA`, the n-th argument positions *for all n*. |
| parsed_doc | The result of `parse_tex(tex_lines)`. |

## minimal_bib *Generate a minimal bibliography file*

### Description

Generate a minimal bibliography file

### Usage

```
minimal_bib(path = ".", bbl.file = NULL, bib.files = NULL, out.bib = bib.files)
```

### Arguments

| | |
|---|---|
| path | A directory containing a document after it has been run with pdflatex. |
| bbl.file | A .bbl file. |
| bib.files | The .bib file or files that were used by BibLaTeX to produce the bibliography. If NULL, the default, the files are inferred from the contents of \\addbibresource within the (unique) .tex file are used. |
| out.bib | The new file of bibliography. |

## parse_tex *Parse LaTeX lines*

### Description

Parse LaTeX lines

### Usage

```
parse_tex(tex_lines)
```

### Arguments

| | |
|---|---|
| tex_lines | Character vector (as read from a .tex file). |

### Value

A data.table where each row identifies a unique character in tex_lines.

line_no Matches the index of tex_lines.

char_no The character within line_no.

char The character. A single character.

tex_group The TeX group by default. Any delimiters can be used.

optional_tex_group (If any present), the optional TeX group.

`tgi` The number of braces opened at the `i`-th current TeX group level.

`GROUP_IDi` An integer identifying the unique contiguous block at the TeX group at or above the current group level.

`GROUP_IDi` The analog for optional groups.

If `tex_lines` is zero-length, a null `data.table`.

## Examples

```
parse_tex(c("A{}", "B[a]{b{c}{d}}z"))
# The version transposed:
#
#>          char : A{}B[a]{b{c}{d}}z
#>           tg1 : 011111122......22
#>           tg2 : 00000000011122222
#>           og1 : 00001111111111111
#>     GROUP_ID1 : .11....222222222.
#>     GROUP_ID2 : .........111222..
#> OPT_GROUP_ID1 : ....111..........
```

---

`position_of_string` *Position of strings*

---

## Description

Position of strings

## Usage

```
position_of_string(tex_line_split, command_split, end = TRUE)

positions_of_all_strings(tex_line, command_name, end = TRUE)
```

## Arguments

| | |
|---|---|
| `tex_line_split` | A split line (via `strsplit(x, split = "")`). |
| `command_split` | The string the position of which is desired, split (via `strsplit(x, split = "")`). |
| `end` | (logical) Should the position of the **end** of the string. By default, `TRUE`; otherwise, the start of the string is chosen. |
| `tex_line` | A line of text. |
| `command_name` | The string the position of which is desired. |

## Value

The end (or start if end = FALSE) of the location of command

## read_tex_document *Read a LaTeX document*

### Description

Read a LaTeX document

### Usage

```
read_tex_document(file_root)
```

### Arguments

| | |
|---|---|
| file_root | The root of the TeX file. |

## report_error *Report errors to console*

### Description

Report errors to console

### Usage

```
report2console(
  file = NULL,
  line_no = NULL,
  column = NULL,
  context = NULL,
  error_message = NULL,
  advice = NULL,
  build_status = NULL,
  extra_cat_ante = NULL,
  extra_cat_post = NULL,
  caret = FALSE,
  rstudio = FALSE,
  log_file = NULL,
  log_file_sep = "|",
  silent = FALSE,
  halt = getOption("TeXCheckR.halt_on_error", FALSE),
  as_tbl = getOption("TeXCheckR.error_as_tbl", FALSE)
)
```

### Arguments

| | |
|---|---|
| file | The file in which the error occurred. |
| line_no | The line number locating the source of the error. |
| column | The position on the line to identify the error (usually following the error). |
| context | The content of the file, to provide context to the error. |
| error_message | The error message to display beyond the console. |
| advice | Advice to the user: how should the detected error be resolved in general? |
| build_status | What should the build status be reported as? |
| extra_cat_ante | Character vector extra messages (placed before context). |
| extra_cat_post | Character vector extra messages (placed after context). |
| caret | (logical, default: FALSE) Should a caret symbol be placed beneath the context to point to the location of the error? The caret will be inserted on a new line after error_message and extra_cat_post.<br>Length-one integer values of caret are permitted and will be interpreted as the number of caret symbols to be inserted at the position. |
| rstudio | If available, should the report be allowed to modify the RStudio session (for example, to pop to the location of the error)? |
| log_file | Optionally, path to a log file on which error_message will be written. |
| log_file_sep | How should the log file's fields be separated? By default, with a pipe (as tabs are common within error messages). |
| silent | (logical, default: FALSE) Suppress all output. |
| halt | Should failures halt via stop or just display a message in the console? |
| as_tbl | Return a list. Experimental. |

---

## rm_editorial_square_brackets

*Remove editorial square brackets*

---

### Description

Change text such as phas[e] out to phase out, without removing square brackets denoting optional arguments.

### Usage

```
rm_editorial_square_brackets(tex_lines)
```

### Arguments

| | |
|---|---|
| tex_lines | Lines (as from readLines). |

## Examples

```
x <- "the BCA's call to `urgently phas[e] out all side deals'"
rm_editorial_square_brackets(x)
```

---

`separate_sentences` *Put sentences on their own line*

---

## Description

Put sentences on their own line

## Usage

```
separate_sentences(filename, hanging_footnotes = FALSE)
```

## Arguments

| | |
|---|---|
| `filename` | A tex or knitr file in which to separate sentences. |
| `hanging_footnotes` | (logical, default: `FALSE`) Should footnotes be indented? |

## Value

NULL. The function is called for its side-effect: rewriting `filename` with separated sentences.

---

`split_report` *Split report into include-able files*

---

## Description

Split report into include-able files

## Usage

```
split_report(
  Report.tex,
  include = TRUE,
  subdir = "tex",
  use.chapter.title = TRUE,
  out.tex = Report.tex
)
```

### Arguments

| | |
|---|---|
| Report.tex | File to split. |
| include | Should \include or \input be used? If TRUE, the default, \include is used. |
| subdir | What directory should each chapter file be written in? By default, a subdirectory of the folder containing Report.tex, called tex, is used. |
| use.chapter.title | Should the chapter title be used to name the chapter files? If TRUE, the default, the title is used (with characters outside [a-zA-Z0-9] replaced by spaces), prefixed by the chapter number; otherwise, just the chapter number is used. |
| out.tex | The new root file. By default, same as Report.tex. |

---

## strip_comments *Strip comments from LaTeX lines*

---

### Description

Strip comments from LaTeX lines

### Usage

```
strip_comments(lines, retain.percent.symbol = TRUE)
```

### Arguments

| | |
|---|---|
| lines | Character vector of a LaTeX document. |
| retain.percent.symbol | (logical, default: TRUE) Should the % symbol itself be stripped? |

### Value

lines but with all text to the right of every unescaped % removed

### Examples

```
some_lines <- c("Text. % A comment", "20\\% of comments are % useful")
strip_comments(some_lines)
strip_comments(some_lines, retain.percent.symbol = FALSE)
```

## tex_group_by_char *TeX group by character position*

### Description

Opening a brace increases the 'group' in TeX. For example, in `a{bc}{d{e}}` `a` is in group 0, `bc` in group 1 as is `d` and `e` is in group 2.

### Usage

```
tex_group_by_char(tex_lines, optional = FALSE)
```

### Arguments

| | |
|---|---|
| `tex_lines` | Character vector of a document LaTeX. |
| `optional` | If `FALSE` (the default), the groups are taken with respect to braces. If `TRUE`, square brackets are used (perhaps not associated with a command). |

### Value

A list the same length as `lines`. Each element an integer vector indicating the TeX group at that position.

For positions **at** braces the **upcoming** group is returned. So `a{b}` should return `0 1 1 0` (in its first element).

### Examples

```
tex_group_by_char("a{bc}{d{e}}")
```

## validate_bibliography *Validate bibliography according to Grattan style*

### Description

Validate bibliography according to Grattan style

### Usage

```
validate_bibliography(path = ".", file = NULL, .report_error, rstudio = FALSE)
```

### Arguments

| | |
|---|---|
| `path` | Containing the bib file. |
| `file` | The bib file if specified. |
| `.report_error` | How errors should be reported. |
| `rstudio` | Use the RStudio API to jump to errors. |

## Details

This is a highly fastidious test of the bibliography. Useful for collaboration to ensure consistent style.

## Value

NULL if bibliography validated.

## Examples

```
## Not run:
bib_temp <- tempfile(fileext = ".bib")
url_bib <-
  paste0("https://raw.githubusercontent.com/HughParsonage/",
         "grattex/e6cab97145d38890e44e83d122e995e3b8936fc6",
         "/bib/Grattan-Master-Bibliography.bib")

download.file(url_bib, destfile = bib_temp)
validate_bibliography(file = bib_temp)

bib_temp <- tempfile(fileext = ".bib")
url_bib <-
  paste0("https://raw.githubusercontent.com/HughParsonage/",
         "grattex/8f7f52a28789d12a363ceb30cea3b41f590ae58a",
         "/bib/Grattan-Master-Bibliography.bib")
download.file(url_bib, destfile = bib_temp)
validate_bibliography(file = bib_temp)

## End(Not run)
```

---

# valid_English_contractions

*Valid English contractions*

---

## Description

List of words which should never raise a spelling error.

## Usage

```
valid_English_contractions
```

## Format

An object of class character of length 110.

### Source

https://gist.githubusercontent.com/J3RN/ed7b420a6ea1d5bd6d06/raw/acda66b325a2b4d7282fb602a7551912cd
contractions.txt

---

## veto_sic *Veto sic*

---

### Description

Vetoes words in a LaTeX document that are marked '[sic]' for the purpose of spell checking by replacing them (and '[sic]' itself) with white space of equal length.

### Usage

```
veto_sic(tex_lines, quote = TRUE, sentence = !quote, words_ante = 1L)
```

### Arguments

| | |
|---|---|
| tex_lines | A character vector. |
| quote | (logical, default: TRUE) Veto words after the previous opening quote (*i.e.* backtick) symbol. |
| sentence | (logical, default: TRUE) Veto words before [sic] in the same sentence. (The start of a sentence is taken to be the location of the capital letter which is preceded by white space and a full stop.) |
| words_ante | The number of words to exclude. Ignored if quote or sentence is TRUE. |

---

## weld_bmillion *Unbreaking spaces between billion and million*

---

### Description

Unbreaking spaces between billion and million

### Usage

```
weld_bmillion(filename, outfile = filename)
```

### Arguments

| | |
|---|---|
| filename | A LaTeX or knitr file. |
| outfile | The file to write to, defaults to filename. |

### Value

NULL. This function is called for its side-effect: rewriting filename with 30 million changed to 30~million.

`wrongly_spelled_words` *List of wrongly spelled words*

## Description

List of wrongly spelled words

## Usage

```
wrongly_spelled_words
```

## Format

A regex of patterns to raise as spelling errors.

# Index


∗ **datasets**
correctly_spelled_words, 16
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE, 16
valid_English_contractions, 29
wrongly_spelled_words, 31

any_bib_duplicates, 3
argument_parsing, 4

bib2DT *(*bib_parser*)*, 5
bib_parser, 5
braces_closes_at, 6

check_biber, 7
check_consecutive_words, 7
check_dashes, 8
check_escapes, 9
check_footnote_typography, 9
check_labels, 10, *19*
check_literal_citations, 11
check_literal_xrefs, 11
check_quote_marks, 12
check_spelling, 12
check_xrefs, 15
commands_used, 15
correctly_spelled_words, *14*, 16
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE, 16

extract_LaTeX_argument, 17
extract_mandatory_LaTeX_argument, *17*, 17
extract_optional_LaTeX_argument, *17*, 18
extract_valid_abbrevations, *14*
extract_valid_abbrevations *(*extract_validate_abbreviations*)*, 19
extract_validate_abbreviations, 19

figs_tbls_unrefd, 19

fread_bib *(*bib_parser*)*, 5

hunspell, *14*

inputs_of, 20
isR_line_in_knitr, 20

lint_bib, *3*, 21
locate_mandatory_LaTeX_argument, 21

md5sum, *8*
minimal_bib, 22

nth_arg_positions *(*argument_parsing*)*, 4

parse_tex, *18*, 22
position_of_string, 23
positions_of_all_strings *(*position_of_string*)*, 23

read_tex_document, 24
reorder_bib *(*bib_parser*)*, 5
replace_nth_LaTeX_argument *(*argument_parsing*)*, 4
report2console, *6*, *13*
report2console *(*report_error*)*, 24
report_error, 24
rm_editorial_square_brackets, 25

separate_sentences, 26
split_report, 26
strip_comments, 27

tex_group_by_char, 28
TeXCheckR-package, 3

valid_English_contractions, 29
validate_bibliography, 28
veto_sic, 30

weld_bmillion, 30
wrongly_spelled_words, 31