Package 'TeXCheckR'

November 5, 2017

Type Package

Title Parses LaTeX Documents for Errors

Date 2017-11-03

Version 0.4.4

Author Hugh Parsonage

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

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 (>= 2.10)

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

LazyData TRUE

RoxygenNote 6.0.1

Suggests testthat, devtools

NeedsCompilation no

Repository CRAN

Date/Publication 2017-11-04 23:38:00 UTC

R topics documented:

TeXCheckR-package 2
any_bib_duplicates 3
argument_parsing 3
bib_parser 4
braces_closes_at 5

check_biber . . . 5
check_consecutive_words . . . 6
check_dashes . . . 6
check_escapes . . . 7
check_footnote_typography . . . 7
check_labels . . . 8
check_literal_citations . . . 9
check_literal_xrefs . . . 9
check_quote_marks . . . 10
check_spelling . . . 10
check_xrefs . . . 12
commands_used . . . 12
correctly_spelled_words . . . 13
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE . . . 13
extract_LaTeX_argument . . . 14
extract_mandatory_LaTeX_argument . . . 14
extract_optional_LaTeX_argument . . . 15
extract_validate_abbreviations . . . 15
figs_tbls_unrefd . . . 16
inputs_of . . . 16
isR_line_in_knitr . . . 17
lint_bib . . . 17
parse_tex . . . 18
position_of_string . . . 18
report_error . . . 19
rm_editorial_square_brackets . . . 20
separate_sentences . . . 20
tex_group_by_char . . . 21
validate_bibliography . . . 21
valid_English_contractions . . . 22
weld_bmillion . . . 23
wrongly_spelled_words . . . 23

Index **24**

`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)
```

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.

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)

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.
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)
```

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.

Value

An error if words are repeated on consecutive lines, together with cat() output of the offending lines. Lastly the `tools::md5sum` of the file is returned in the error message, so it can be supplied to `md5sum.ok`. `NULL` otherwise.

check_dashes *Check dashes entered as hyphens*

Description

Check dashes entered as hyphens

Usage

```
check_dashes(filename, .report_error)
```

Arguments

filename	A tex or Rnw file.
.report_error	How errors should be reported.

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)
```

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.

Details

See https://github.com/HughParsonage/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)
```

Arguments

`filename`	The LaTeX source file to check.
`.report_error`	The function to provide context to the error.

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, pre_release = TRUE, ignore.lines = NULL,
  known.correct = NULL, known.wrong = NULL, bib_files, check_etcs = TRUE,
  dict_lang = "en_GB", rstudio = FALSE, .report_error)
```

Arguments

filename	Path to a LaTeX file to check.
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.wrong	Character vector of patterns known to be wrong.
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.

Details

Extends and enhances hunspell. The advantage of this function is that 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. You can also add the directive % ignore_spelling_in: mycmd which will ignore the spelling of words within the first argument of \mycmd.

Another possible advantage is that only the root document need be supplied; any files that are fed via \input or \include are checked (recursively).

Other advantages included skipping the contents of certain commands, the spelling of which need not be checked as they are not printed, viz. citation and cross-reference commands, and certain optional arguments. These have been fixed by hunspell since the first version of this package.

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/HughParsonage/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 incorrect case.

Usage

```
check_xrefs(filename, .report_error)
```

Arguments

filename	A LaTeX file
.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.

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`). If not supplied (the default) the contents of `tex_lines` are passed through `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.

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.

GRP_ID An integer identifying the unique contiguous block.

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

`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,
  rstudio = FALSE, log_file = NULL)
```

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`).
`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.

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)
```

Arguments

`filename`	A tex or knitr file in which to separate sentences.

Value

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

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

`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

∗Topic **datasets**
correctly_spelled_words, 13
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE, 13
valid_English_contractions, 22
wrongly_spelled_words, 23

any_bib_duplicates, 3
argument_parsing, 3

bib2DT *(*bib_parser*)*, 4
bib_parser, 4
braces_closes_at, 5

check_biber, 5
check_consecutive_words, 6
check_dashes, 6
check_escapes, 7
check_footnote_typography, 7
check_labels, 8, *16*
check_literal_citations, 9
check_literal_xrefs, 9
check_quote_marks, 10
check_spelling, 10
check_xrefs, 12
commands_used, 12
correctly_spelled_words, *11*, 13
CORRECTLY_SPELLED_WORDS_CASE_SENSITIVE, 13

extract_LaTeX_argument, 14
extract_mandatory_LaTeX_argument, *14*, 14
extract_optional_LaTeX_argument, *14*, 15
extract_validate_abbreviations, 15

figs_tbls_unrefd, 16
fread_bib *(*bib_parser*)*, 4

inputs_of, 16
isR_line_in_knitr, 17

lint_bib, *3*, 17

nth_arg_positions *(*argument_parsing*)*, 3

parse_tex, *14*, 18
position_of_string, 18
positions_of_all_strings *(*position_of_string*)*, 18

reorder_bib *(*bib_parser*)*, 4
replace_nth_LaTeX_argument *(*argument_parsing*)*, 3
report2console *(*report_error*)*, 19
report_error, 19
rm_editorial_square_brackets, 20

separate_sentences, 20

tex_group_by_char, 21
TeXCheckR-package, 2

valid_English_contractions, 22
validate_bibliography, 21

weld_bmillion, 23
wrongly_spelled_words, 23