

# Package ‘parseLatex’

July 23, 2025

**Type** Package

**Title** Parse 'LaTeX' Code

**Version** 0.4.1

**Description** Exports an enhanced version of the tools::parseLatex() function to handle 'LaTeX' syntax more accurately. Also includes numerous functions for searching and modifying 'LaTeX' source.

**License** GPL (>= 2)

**Encoding** UTF-8

**URL** <https://github.com/dmurdoch/parseLatex>,  
<https://dmurdoch.github.io/parseLatex/>

**BugReports** <https://github.com/dmurdoch/parseLatex/issues>

**Suggests** kableExtra, knitr, rmarkdown, testthat (>= 3.0.0)

**Imports** utils

**Depends** R (>= 4.1.0)

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**SystemRequirements** gettext, bison (>= 3.0)

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Duncan Murdoch [aut, cre],  
The R Core Team [ctb, cph]

**Maintainer** Duncan Murdoch <murdoch.duncan@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-06-06 17:00:05 UTC

## Contents

as_LaTeX2	2
defaultCatcodes	3
deparseLatex	4
finders	4
find_caption	6
find_pattern	7
find_sequence	8
find_tableContent	9
find_tableRow	10
get_contents	11
get_leftovers	12
itemlist	12
LaTeX2range	13
names	14
options	15
parseLatex_fn	16
parseLatex_pkg	18
path_to	18
path_to_index	20
prepare_table	21
print.LaTeX2item	22
reduce_whitespace	22
rmSrcrefs	23
set_range	23
showErrors	24
splitting	25
tablecalcs	26
tableCell	26
tableOption	28
tableRule	29
tables	31
tests	32
Utilities	33
vector_to_latex2	34
vector_to_row	35
<b>Index</b>	<b>36</b>

---

as\_LaTeX2

*Coerce to LaTeX2*


---

### Description

Coerce to LaTeX2

**Usage**

```
as_LaTeX2(x)
```

```
latex2(...)
```

**Arguments**

x                    An object to convert to a [LaTeX2](#) object.

...                  Objects to concatenate.

**Value**

as\_LaTeX2() converts x to a [LaTeX2](#) object.

latex2() converts the arguments to [LaTeX2](#) objects and concatenates them into a new [LaTeX2](#) object.

---

defaultCatcodes	<i>The default "catcodes" used by <a href="#">parseLatex</a>.</i>
-----------------	---

---

**Description**

The default "catcodes" used by [parseLatex](#).

**Usage**

```
defaultCatcodes
```

**Format**

An object of class `data.frame` with 13 rows and 2 columns.

**Details**

defaultCatcodes is a dataframe containing the default catcode definitions. The numeric values of each code are exported, e.g. LETTER is 11.

**Examples**

```
# \makeatletter has no effect by default...
unclass(parseLatex("\makeatletter\internal@macro"))
# ... but the effect can be simulated
atletter <- rbind(defaultCatcodes,
                  data.frame(char="@", catcode=11))
unclass(parseLatex("\makeatletter\internal@macro",
                  catcodes = atletter))
# These are the default codes:
cbind(defaultCatcodes, name = c("ESCAPE", "LBRACE", "RBRACE", "MATH",
                              "ALIGN", "NEWLINE", "NEWLINE", "PARAM", "SUPER",
```

```

    "SUB", "SPACE", "SPACE", "COMMENT"))
# The missing ones are
# 9 - IGNORE
# 11 - LETTER
# 12 - OTHER
# 13 - ACTIVE
# 15 - INVALID

```

---

deparseLatex                      *Convert latex object into character vector*

---

### Description

Convert latex object into character vector

### Usage

```
deparseLatex(x, dropBraces = FALSE)
```

### Arguments

x                      A latex object.  
dropBraces            Whether to drop unnecessary braces.

### Value

deparseLatex returns character vector corresponding to the parsed Latex.

---

finders                      *Miscellaneous low-level finders*

---

### Description

Miscellaneous low-level finders

### Usage

```

find_whitespace(items, ...)

find_env(items, envtypes = NULL, ...)

find_macro(items, macros = NULL, ...)

find_catcode(items, codes, ...)

find_tags(items, tags, ...)

```

```
find_char(items, char, ...)
```

```
find_block(items, ...)
```

```
find_general(items, test, ..., all = TRUE, path = FALSE)
```

### Arguments

<code>items</code>	A list of latex items.
<code>...</code>	For <code>find_general</code> , additional arguments to pass to <code>test</code> . For the other <code>find_*</code> functions, additional arguments to pass to <code>find_general</code> .
<code>envtypes</code>	Which types of environment to look for.
<code>macros</code>	Which types of macros to look for.
<code>codes</code>	Which codes to look for.
<code>tags</code>	Which tags to look for.
<code>char</code>	Which character to look for.
<code>test</code>	Test function for target.
<code>all</code>	If FALSE, return just the first match
<code>path</code>	If TRUE, return a path rather than an index. See Details below.

### Details

These functions search through `items` for individual objects that match a test. In general they do not operate recursively, with one exception. If `items` contains `ITEMLIST` objects, the search will always recurse into those.

By default the return value is an index or a vector of indices of the matches. These are the indices as they would be if any `ITEMLIST` objects had been flattened.

However, if `path = TRUE`, the path to the object will be returned. With `all = FALSE`, this will be a numeric vector such that `items[[result]]` is the matching item. With `all = TRUE` it will be a list of such vectors.

### Value

`find_whitespace()` returns the indices of whitespace in `items`.

`find_env()` returns the indices within `items` of environments in `envtypes`.

`find_macro()` returns the index within `items` of instances in `macros`.

`find_catcode()` returns the index within `items`. of specials matching code.

`find_tags()` returns the index within `items`. of items with tags matching `tags`.

`find_char()` returns the index within `items` of characters matching `char`. Only characters marked as `SPECIAL` by the parser will be found.

`find_block()` returns the index within `items` of blocks (i.e. sequences in )

`find_general()` returns locations of objects matching the test.

**See Also**

[index\\_to\\_path\(\)](#), [path\\_to\\_index\(\)](#)

---

find_caption	<i>Find or drop captions</i>
--------------	------------------------------

---

**Description**

Find or drop captions

**Usage**

```
find_caption(items)
```

```
drop_caption(items, idx = NULL)
```

```
path_to_caption(items)
```

**Arguments**

items            A [LaTeX2](#) or other list of [LaTeX2items](#).

idx              NULL or a vector of the same length as items

**Value**

find\_caption() returns a [LaTeX2range](#) object for any caption text, with an attribute extra holding the range of associated macros and whitespace.

drop\_caption() returns the items with captions dropped as a [LaTeX2](#) object. It has an attribute named idx that is the idx argument with corresponding elements dropped.

path\_to\_caption() returns a path containing the location of the first caption block within items. It has an attribute extra containing a [LaTeX2range](#) object for the associated macros and whitespace.

**Examples**

```
parsed <- parseLatex("before \\caption{This is a caption} \\\ after")
idx <- find_caption(parsed)
get_range(parsed, idx)
get_range(parsed, attr(idx, "extra"))
drop_caption(parsed)
path_to_caption(parsed)
```

---

find_pattern	<i>Find a pattern in deparsed items</i>
--------------	---

---

## Description

Searches a [LaTeX2](#) list for text using `grepl()` on deparsed versions of parts of the code. It attempts to find the narrowest match(es) that lie within a single container.

## Usage

```
find_pattern(items, pattern, ..., all = FALSE)
```

## Arguments

<code>items</code>	A list of latex items.
<code>pattern</code>	Pattern to use in <code>grepl()</code> .
<code>...</code>	Additional parameters to pass to <code>grepl</code> .
<code>all</code>	Find all matching, or the first?

## Details

`find_pattern()` does a recursive search in the order items appear in the deparse. If the pattern matches, it attempts to narrow the match by recursing into containers and dropping earlier and later items. It should always return syntactically correct LaTeX code in which the pattern appears.

## Value

`find_pattern()` returns a [LaTeX2range](#) object or (if `all` is `TRUE`) a list of them.

## Examples

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex", caption = "Sample table")
parsed <- parseLatex(latex)
parsed
loc <- find_pattern(parsed, "RX4 Wag", fixed = TRUE)
loc
print(loc, source = parsed)
```

---

find_sequence	<i>Find a code sequence</i>
---------------	-----------------------------

---

### Description

Find a code sequence

### Usage

```
find_sequence(items, sequence, all = FALSE, ignore_whitespace = TRUE)
```

```
items_are_equal(items1, items2)
```

### Arguments

items, sequence [LaTeX2](#) objects or lists.

all Whether to return all matches, or just the first.

ignore\_whitespace

Whether to ignore whitespace in comparisons.

items1, items2 Two [LaTeX2](#) or [LaTeX2item](#) objects.

### Details

`find_sequence()` will only match sequences that are entirely contained within a single list in `items`. Thus if `prepare_table()` is called on a table, then `find_sequence()` will find sequences in the code before or after the rows, or entirely within a single cell, but not crossing alignment markers ("&").

### Value

`find_sequence()` returns a [LaTeX2range](#) or list of them where sequence occurs within `items`.

`items_are_equal` returns a logical indicator of equality after removing source references.

### Examples

```
find_sequence(parseLatex("a & b & c"), "b & c")
```



---

find\_tableContent      *Functions relating to the data content of a table*

---

### Description

Functions relating to the data content of a table

### Usage

```
find_tableContent(table)
```

```
tableContent(table)
```

```
tableContent(table, asis = FALSE) <- value
```

### Arguments

table	A tabular-like environment to work with. It must not be one for which <code>prepare_table()</code> has been called.
asis	Should newlines be added around the value?
value	The content to be inserted into the cell. This can be a <a href="#">LaTeX2</a> object, or a character string that will be converted to one.

### Details

Unless `asis = TRUE`, `tableContent(table) <- value` will add newlines at the start end end if not present, to make the result more readable.

### Value

`find_tableContent()` returns the indices of the entries corresponding to content of the table.

`tableContent()` returns a [LaTeX2](#) object containing all of the table content after the options.

### Examples

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
table
tableContent(table)

tableContent(table) <- "Mazda RX4 & 21 & 6\\\\"
table
tableContent(table, asis = TRUE) <- "Mazda RX4 & 21 & 6\\\\"
table
```

---

`find_tableRow`*Functions to work with rows in tables*

---

## Description

Functions to work with rows in tables

## Usage

```
find_tableRow(table, row, withExtras = FALSE, withData = TRUE)
```

```
tableRow(table, row, withExtras = FALSE, withData = TRUE)
```

```
tableRow(table, row, asis = FALSE, withExtras = FALSE, withData = TRUE) <- value
```

## Arguments

<code>table</code>	A tabular-like environment to work with.
<code>row</code>	row in the table (1 is top row), including rows of labels.
<code>withExtras</code>	If TRUE, include the extras before the line of data, such as <code>\hline</code> , etc.
<code>withData</code>	If TRUE, include the data.
<code>asis</code>	Should a linebreak and newline be added after the value?
<code>value</code>	The content to be inserted into the cell. This can be a <a href="#">LaTeX2</a> object, or a character string that will be converted to one.

## Details

Unless `asis = TRUE`, `tableContent(table) <- value` will add `"\n"` and a newline at the end if not present.

If the row value is higher than the number of rows in the table, blank rows will be added to fill the space between.

If `withExtras = TRUE` and you want the result to start on a new line, you need to add the newline explicitly in `value` when using the assignment function.

## Value

`find_tableRow()` returns a [LaTeX2range](#) of the entries corresponding to the content of row `i` of the table.

`tableRow()` returns a [LaTeX2](#) object containing all of the table content in the row.

**Examples**

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
find_tableRow(table, 1)

tableRow(table, 1)
tableRow(table, 1, withExtras = TRUE)

tableRow(table, 5) <- "a & b & c"
table
```

---

get\_contents

*Convenience functions to get or set contents of item*

---

**Description**

Convenience functions to get or set contents of item

**Usage**

```
get_contents(item)

set_contents(item, value)
```

**Arguments**

`item` An item from a Latex list (or a [LaTeX2](#) list with one item).

`value` An object that can be coerced to be a [LaTeX2](#) object.

**Value**

`get_contents` returns the contents of the item as a [LaTeX2](#) list.

`set_contents` returns the original `item` with the contents replaced by `value`.

**Examples**

```
get_contents(parseLatex("{abc}"))

set_contents(parseLatex("{abc}"), "def")
```

---

<code>get_leftovers</code>	<i>Retrieve source from beyond the end of the document.</i>
----------------------------	---

---

**Description**

Retrieve source from beyond the end of the document.

**Usage**

```
get_leftovers(text, items = parseLatex(text))
```

**Arguments**

<code>text</code>	Character vector holding source.
<code>items</code>	Parsed version of text.

**Value**

The part of `text` that follows `\end{document}` other than a single newline, named according to the original line numbers.

**Note**

The line numbering in the output matches what a text editor would see; embedded newlines in `text` will result in separate lines in the output.

**Examples**

```
# line: 1           2           3
text <- "\\begin{document}\n\\end{document}\nnotes"
get_leftovers(text)
```

---

<code>itemlist</code>	<i>Lists of items</i>
-----------------------	-----------------------

---

**Description**

Lists of items

**Usage**

```
new_itemlist(...)

flatten_itemlists(items, recursive = FALSE)

placeholder()

show_itemlists(items, indent = 0, verbose = FALSE)
```

**Arguments**

...	Items to be passed to latex2().
items	A list of <a href="#">LaTeX2item</a> objects.
recursive	Whether to proceed recursively.
indent	How much to indent the display?
verbose	Whether to show tags of non-itemlists and details of each itemlist.

**Details**

An ITEMLIST is a list of items. Deparsing it just concatenates the parts. This is intended to be used when parsing tables, for example, where it makes sense to break up the table into individual rows. See [prepare\\_table](#) for more details.

**Value**

`new_itemlist()` returns an ITEMLIST item containing the items.

`flatten_itemlists()` returns items with ITEMLIST items expanded. If items itself was an ITEMLIST, it is returned as a [LaTeX2](#) object; otherwise its type will be unchanged. The result will never include any ITEMLIST or PLACEHOLDER items at the top level, and if recursive is TRUE, not at any level.

`placeholder()` returns a LaTeX2item object with tag PLACEHOLDER. These will never print, and are used as spacers within an ITEMLIST.

`show_itemlists()` is a debugging function called for the side effect of displaying the itemlist structure of an object.

**Examples**

```
new_itemlist(parseLatex("abc def"), label = "items")
```

---

LaTeX2range

*Ranges within LaTeX2 lists.*


---

**Description**

Ranges within LaTeX2 lists.

**Usage**

```
LaTeX2range(path, range)
```

```
## S3 method for class 'LaTeX2range'
print(x, source = NULL, ...)
```

**Arguments**

path	An integer vector to use as a path.
range	A range of values within the path.
x	Object to print.
source	Optional parsed list from which to extract the range.
...	Ignored.

**Details**

LaTeX2range objects are lists with path and range entries. path is a recursive index into a [LaTeX2](#) list, and range is a range of entries in the result.

If path is NULL, the object refers to the entire source object. If range is NULL, it refers to the whole [LaTeX2item](#) given by the path.

**Value**

LaTeX2range() returns a constructed LaTeX2range object.

---

names

*Utility functions finding names and types of objects*

---

**Description**

Utility functions finding names and types of objects

**Usage**

```
latexTag(item)
```

```
catcode(item)
```

```
envName(item)
```

```
envName(item) <- value
```

```
macroName(item)
```

**Arguments**

item	A <a href="#">LaTeX2item</a> which is an environment
value	A character string to set as the name

**Value**

`latexTag()` returns the [LaTeX2](#) tag for the item or NULL.

`catcode()` returns the TeX catcode for the item, or NULL.

`envName()` returns the Latex environment name for an item, or NULL.

`macroName()` returns the Latex macro, or NULL.

---

options

*Find or modify macro or environment options*

---

**Description**

Many Latex environments and macros take optional parameters wrapped in square brackets. `find_bracket_options` finds those, assuming they come immediately after the macro.

Some Latex environments and macros take optional parameters wrapped in curly brackets (braces). `find_brace_options` finds those if they immediately follow the environment or macro (and possibly some bracketed options).

**Usage**

```
find_bracket_options(items, which = 1L, start = 1L)
```

```
bracket_options(items, which = 1L, start = 1L)
```

```
bracket_options(items, which = 1, start = 1, asis = FALSE) <- value
```

```
find_brace_options(items, which = 1L, start = 1L, path = FALSE)
```

```
brace_options(items, which = 1, start = 1)
```

```
brace_options(items, which = 1, start = 1, asis = FALSE) <- value
```

**Arguments**

`items` A list of latex items.

`which` Which options do you want? Some macros support more than one set.

`start` Start looking at `items[[start]]`. `start` may be a path.

`asis` Should newlines be added around the value?

`value` The content to be inserted into the cell. This can be a [LaTeX2](#) object, or a character string that will be converted to one.

`path` If TRUE, return a path rather than an index, as with `find_general()`,

**Value**

find\_bracket\_options returns a [LaTeX2range](#) object pointing to the options within items (including the brackets).

bracket\_options returns a [LaTeX2](#) object containing the specified options.

find\_brace\_options returns the index or path to the block containing the options.

brace\_options returns a [LaTeX2](#) object containing the specified options.

**Examples**

```

parsed <- parseLatex("\\section[a]{b}")
macro <- find_macro(parsed, "\\section")
bracket_options(parsed, start = macro + 1)

bracket_options(parsed, start = macro + 1) <- "Short Title"
parsed

brace_options(parsed, start = macro + 1)

brace_options(parsed, start = macro + 1) <- "Long Title"
parsed

```

---

 parseLatex\_fn

*Parse LaTeX code*


---

**Description**

The parseLatex function parses LaTeX source, producing a structured object.

**Usage**

```

parseLatex(
  text,
  verbose = FALSE,
  verbatim = c("verbatim", "verbatim*", "Sinput", "Soutput"),
  verb = "\\Sexpr",
  defcmd = c("\\newcommand", "\\renewcommand", "\\providecommand", "\\def",
    "\\let"),
  defenv = c("\\newenvironment", "\\renewenvironment"),
  catcodes = defaultCatcodes,
  recover = FALSE,
  showErrors = recover,
  ...
)

```



**Arguments**

text	A character vector containing LaTeX source code.
verbose	If TRUE, print debug error messages.
verbatim	A character vector containing the names of L <sup>A</sup> T <sub>E</sub> X environments holding verbatim text.
verb	A character vector containing LaTeX macros that should be assumed to hold verbatim text.
defcmd, defenv	Character vectors of macros that are assumed to define new macro commands or environments respectively. See the note below about some limitations.
catcodes	A list or dataframe holding LaTeX "catcodes", such as <a href="#">defaultCatcodes</a> .
recover	If TRUE, attempt to recover from errors and continue parsing. See Details below.
showErrors	If TRUE, show errors after parsing.
...	Additional parameters to pass to <a href="#">showErrors</a> .

**Details**

Some versions of LaTeX such as pdf<sub>l</sub>atex only handle ASCII inputs, while others such as xelatex allow Unicode input. parseLatex allows Unicode input.

During processing of LaTeX input, an interpreter can change the handling of characters as it goes, using the `\catcode` macro or others such as `\makeatletter`. However, parseLatex() is purely a parser, not an interpreter, so it can't do that, but the user can change handling for the whole call using the catcodes argument.

catcodes should be a list or dataframe with at least two columns:

- char should be a column of single characters.
- catcode should be a column of integers in the range 0 to 15 giving the corresponding catcode.

During parsing, parseLatex will check these values first. If the input character doesn't match anything, then it will be categorized:

- as a letter (catcode 11) using the ICU function `u_hasBinaryProperty(c, UCHAR_ALPHABETIC)` (or `iswalph(c)` on Windows),
- as a control character (catcode 15) if its code point is less than 32,
- as "other" (catcode 12) otherwise.

When recover = TRUE, the parser will mark each error in the output, and attempt to continue parsing. This may lead to a cascade of errors, but will sometimes help in locating the first error. The section of text related to the error will be marked as an item with tag ERROR.

**Value**

parseLatex returns parsed Latex in a list with class "LaTeX2". Items in the list have class "LaTeX2item".

**defcmd limitations**

The LaTeX defining commands have fairly simple syntax, but `\def` and `\let` from plain Tex have quite variable syntax and parseLatex() does not attempt to handle it all. Stick with simple syntax like `\def\bea{\begin{eqnarray}}` and it should work.

**See Also**

LaTeX2, LaTeX2item

**Examples**

```
parsed <- parseLatex(r"(fran\c{c}ais)")
parsed
```

---

parseLatex\_pkg      *The parseLatex package*

---

**Description**

Exports an enhanced version of the tools::parseLatex() function to handle 'LaTeX' syntax more accurately. Also includes numerous functions for searching and modifying 'LaTeX' source.

**Author(s)**

**Maintainer:** Duncan Murdoch <murdoch.duncan@gmail.com>

Other contributors:

- The R Core Team [contributor, copyright holder]

**See Also**

Useful links:

- <https://github.com/dmurdoch/parseLatex>
- <https://dmurdoch.github.io/parseLatex/>
- Report bugs at <https://github.com/dmurdoch/parseLatex/issues>

---

path\_to      *Find path to a particular kind of item*

---

**Description**

Find path to a particular kind of item

**Usage**

```

path_to(items, test, ..., all = FALSE)

get_item(items, path = index_to_path(index, items), index)

get_items(items, paths = lapply(indices, index_to_path, items), indices)

set_item(items, path, value)

insert_values(items, path, values, after = FALSE)

get_container(items, path)

get_which(path)

```

**Arguments**

items	A list of latex items.
test	Which test function to use.
...	Additional parameters to pass to <code>is_fn</code> .
all	Return all paths, or just the first?
path	Integer vector of subitems
index	Index into the flattened version of <code>items</code> .
paths	List of paths
indices	Vector of indices into the flattened version of <code>items</code> .
value	A <a href="#">LaTeX2item</a> to set as a value.
values	A <a href="#">LaTeX2</a> list or a <a href="#">LaTeX2item</a> .
after	If TRUE, insert the values after path.

**Details**

`path_to()` does a recursive search in the order items appear in the deparse.

**Value**

`path_to()` returns the recursive path to the first example matching the `is_fn` conditions, or a list of paths to all matching items.

`get_item()` returns the item at the given path. If `index` is specified, `get_item()` will return that item in the flattened version of `items`.

`get_items()` returns the items at the given paths as a [LaTeX2](#) object. If `index` is specified, `get_items()` will return those items in the flattened version of `items`.

`set_item()` replaces the item at the given path, and returns the modified version of `items`.

`insert_values()` inserts the values before the item mentioned in `path` (or after if requested), and returns the modified version of `items`.

`get_container()` returns the item containing the given path  
`get_which()` returns the index of the item within its container.

### Examples

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex", caption = "Sample table")
parsed <- parseLatex(latex)
parsed
path <- path_to(parsed, test = is_env,
                envtypes = "tabular")
get_item(parsed, path)
```

---

path\_to\_index                      *Convert between paths and indices*

---

### Description

Convert between paths and indices

### Usage

```
path_to_index(path, items)

index_to_path(index, items)

paths_to_ranges(path1, path2, items)

get_ranges(items, ranges)
```

### Arguments

path	A vector of integers, assumed to be a path through "ITEXMLIST" entries in a <a href="#">LaTeX2</a> or <a href="#">LaTeX2item</a> object.
items	The referenced object.
index	A scalar integer which would be the index to an item if <code>items</code> was flattened.
path1, path2	Paths into the same destination list.
ranges	A list of <a href="#">LaTeX2range</a> objects, such as that produced by <code>paths_to_ranges()</code> .

### Value

`path_to_index` returns a scalar value corresponding the the index if `items` was flattened.  
`index_to_path` returns a vector of integers which would index the specified item.  
`paths_to_range` returns a list of [LaTeX2range](#) objects covering all entries extending from `path1` to `path2`.  
`get_ranges()` extracts the specified ranges, concatenates them, and returns them as a [LaTeX2](#) object.

**See Also**

[flatten\\_itemlists\(\)](#)

**Examples**

```

latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
tablepath <- path_to(parsed, is_env, envtypes = "tabular")
table <- prepare_table(parsed[[tablepath]])
path_to_index(c(4,1,1), table)
index_to_path(3, table)

ranges <- paths_to_ranges(index_to_path(3, table),
                          c(4,1,1), table)
lapply(ranges, get_range, items = table)

get_ranges(table, ranges)

```

---

prepare_table	<i>Split up a table by rows</i>
---------------	---------------------------------

---

**Description**

Split up a table by rows

**Usage**

```

prepare_table(table, do_cells = TRUE)

prepare_row(row)

```

**Arguments**

table	A tabular-like environment to work with.
do_cells	Should the rows be prepared too?
row	A list of items from a single row of a table.

**Value**

A [LaTeX2item](#) object which is the same table but with the contents divided into [ITEMLIST](#)s. The first element is an [ITEMLIST](#) holding everything before the first row, then each row is in its own [ITEMLIST](#), and finally one more holding everything after the last row. The attribute `has_itemlists` will be set to `TRUE`.

`prepare_row()` returns a [LaTeX2item](#) object which is the same row with [ITEMLIST](#)s holding the cells. The attribute `has_itemlist` will be set to `TRUE`. The first list will be the "extras" at the start of the row; each cell will be in the following [ITEMLIST](#)s. The following cell delimiter will be included in the cell.

**Examples**

```

latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- prepare_table(parsed[[find_tabular(parsed)]])
print(latex2(table), tags = TRUE)
row <- prepare_row(tableRow(table, 2))
print(latex2(row), tags = TRUE)

```

---

print.LaTeX2item      *Print methods*

---

**Description**

Print methods

**Usage**

```

## S3 method for class 'LaTeX2item'
print(x, ...)

## S3 method for class 'LaTeX2'
print(x, tags = FALSE, ...)

```

**Arguments**

x	Object to work on.
...	Extra parameters to pass to <a href="#">deparseLatex</a> .
tags	Whether to display LaTeX2 tags.

---

reduce\_whitespace      *Remove excess whitespace recursively*

---

**Description**

Remove excess whitespace recursively

**Usage**

```
reduce_whitespace(items, recursive = TRUE, all = FALSE)
```

**Arguments**

items	A <a href="#">LaTeX2</a> object.
recursive	Apply to all lists within items.
all	If TRUE, remove all white space, not just doubles.

**Value**

items with double spaces or double newlines set to single, and trailing spaces removed (or all whitespace removed, if all is TRUE).

**Examples**

```
parsed <- parseLatex("a {b\n\nc}")
parsed
reduce_whitespace(parsed)
```

---

rmSrcrefs	<i>Remove srcrefs</i>
-----------	-----------------------

---

**Description**

Remove srcrefs

**Usage**

```
rmSrcrefs(items)
```

**Arguments**

items            A [LaTeX2](#) object, or any other list of [LaTeX2](#) items.

**Value**

The items with source references removed.

---

set_range	<i>Set items in a <a href="#">LaTeX2</a> object</i>
-----------	---

---

**Description**

Set items in a [LaTeX2](#) object

**Usage**

```
set_range(items, range, values)

get_range(items, range)
```

**Arguments**

items	A <a href="#">LaTeX2</a> object or other list of <a href="#">LaTeX2item</a> objects.
range	A <a href="#">LaTeX2range</a> object.
values	An object that can be coerced to a <a href="#">LaTeX2</a> object or (if range\$range is NULL) a <a href="#">LaTeX2item</a> .

**Value**

set\_range() replaces the item(s) at the given path, and returns the modified version of items.  
 get\_range() extracts the specified range and returns it as a [LaTeX2](#) object.

**Examples**

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex", caption = "Sample table")
parsed <- parseLatex(latex)
tablepath <- path_to(parsed, is_env, envtypes = "tabular")
range <- LaTeX2range(tablepath, 11)
parsed <- set_range(parsed, range, "The 11th item")
parsed
get_range(parsed, range)
```

---

 showErrors

*Show errors in parsed Latex object*


---

**Description**

Show errors in parsed Latex object

**Usage**

```
showErrors(
  x,
  repeatSrcline = FALSE,
  errorMsgTwice = FALSE,
  lineNumbers = TRUE,
  showAllLines = FALSE
)
```

**Arguments**

x	A <a href="#">LaTeX2</a> object.
repeatSrcline	Repeat the source line when it has multiple errors?
errorMsgTwice	Show the error message at both the start and end of a multiline error?
lineNumbers	Show line numbers on output?
showAllLines	Show all lines whether they have errors or not?



**Value**

A list of paths to errors, invisibly.

**Examples**

```
parsed <- parseLatex("\\end{baz} \\begin{foo} \\n \\begin{bar} $1+1\\n4",
                    recover = TRUE, showErrors = FALSE)
showErrors(parsed)
```

---

splitting

*Splitting lists of items*


---

**Description**

Splitting lists of items

**Usage**

```
split_list(items, splits, include = FALSE)

split_latex(...)
```

**Arguments**

items	A <a href="#">LaTeX2</a> or similar list.
splits	Which item numbers divide the parts?
include	If TRUE, include the split item at the end of each part.
...	Arguments to pass to <code>split_list</code> .

**Value**

`split_list()` returns a list of pieces separated at the splits.

`split_latex()` returns a list of pieces separated at the splits. Each piece is marked as an ITEM LIST item, and the whole thing is also marked that way.

---



*Calculations on tables*


---

**Description**

Calculations on tables

**Usage**

```
tableNrow(table)
```

```
tableNcol(table)
```

```
tableDim(table)
```

**Arguments**

table            A known tabular-like environment object.

**Value**

tableNrow() returns the number of rows in the table.

tableNcol() returns the number of columns in the table.

tableDim() returns the number of rows and columns in the table.

**Examples**

```
latex <- kableExtra::kbl(mtcars[1:2, 1:3], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
table
tableNrow(table)
tableNcol(table)
tableDim(table)
```

---

tableCell

*Work with table cells*


---

**Description**

These functions work with the content of cells in tabular-like environments. Cells are numbered with the first row (typically column titles) being row 1. Rules (i.e. horizontal lines) are not considered part of a cell.

**Usage**

```
find_tableCell(table, row, col)

tableCell(table, row, col)

tableCell(table, row, col, asis = FALSE) <- value
```

**Arguments**

table	A tabular-like environment to work with.
row, col	row and column in the table.
asis	Should blanks be added around the value?
value	The content to be inserted into the cell. This can be a <a href="#">LaTeX2</a> object, or a character string that will be converted to one.

**Details**

find\_tableCell() returns NULL if the requested cell is missing because an earlier cell covered multiple columns. It signals an error if a request is made beyond the bounds of the table.

Unless asis = TRUE, tableContent(table) <- value will add blanks at the start end end if not present, to make the result more readable.

If col is higher than the current table width, the assignment will fail with an error. If only row is too high, blank lines will be added and it should succeed.

**Value**

find\_tableCell() returns a [LaTeX2range](#) object giving the location of the requested cell.

tableCell() returns a [LaTeX2](#) object containing all of the table content in the cell (but not the &).

**Examples**

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- prepare_table(parsed[[find_tabular(parsed)]], do_cells = TRUE)
find_tableCell(table, 1, 2)

tableCell(table, 1, 2)

tableCell(table, 5, 2) <- " d "
table
```

---

tableOption	<i>Functions related to table options.</i>
-------------	--

---

### Description

Functions related to table options.

### Usage

```
find_posOption(table)
posOption(table)
posOption(table, asis = FALSE) <- value
find_widthOption(table)
widthOption(table)
widthOption(table, asis = FALSE) <- value
find_columnOptions(table)
columnOptions(table)
columnOption(table, column)
columnOptions(table, asis = FALSE) <- value
columnOption(table, column) <- value
```

### Arguments

table	A known tabular-like environment object, or the contents of one.
asis	Whether to make small modifications in replacement functions.
value	A character string or <a href="#">LaTeX2</a> object.
column	For which column?

### Details

Unless `asis == TRUE`, the value for `value` in `posOption(table) <- value` can be specified with or without the enclosing brackets.

**Value**

find\_posOption() returns the indices of the entries corresponding to the "pos" option, including the brackets, within the table.

posOption() returns a [LaTeX2](#) object containing the "pos" option.

find\_widthOption() returns the index of the block corresponding to the "width" option, if there is one. Only some tabular-like environments have these.

widthOption() returns a [LaTeX2](#) object containing the "width" option, if the table has one.

find\_columnOptions() returns a [LaTeX2range](#) object for the column options of the table.

columnOptions() returns a [LaTeX2](#) object containing the "column" options.

columnOption() returns a [LaTeX2](#) object containing the requested column option. A "|" divider will not be included.

**Examples**

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
table
find_posOption(table)

posOption(table)

posOption(table) <- "h"
posOption(table)
find_widthOption(table)

widthOption(table)

find_columnOptions(table)
columnOptions(table)

columnOption(table, 3)
columnOptions(table) <- "lrr"
table
columnOption(table, 3) <- "p{1cm}"
columnOptions(table)
```

---

tableRule

*Work with rules in tables*


---

**Description**

In LaTeX, "rules" are horizontal lines in a table. These functions let rules be extracted or modified.

**Usage**

```

find_rules(table)

rules(table, idx = find_rules(table))

find_rule(table, row)

rule(table, row)

rule(table, row, asis = FALSE, idx = find_rules(table)) <- value

```

**Arguments**

table	A tabular-like environment to work with.
idx	A list of indices as produced by <code>find_rules()</code> .
row	The rules will precede the contents of this row. The rule after the final row uses <code>row = tableNrow(table) + 1</code> .
asis	Should a newline be added after the value? If <code>asis = TRUE</code> , it will not be.
value	The content to be inserted into the cell. This can be a <a href="#">LaTeX2</a> object, or a character string that will be converted to one.

**Value**

`find_rules()` returns a list of [LaTeX2range](#) objects giving the locations of the rules before each line. The last item in the list gives the location of any rules after the last line.

`rules(table)` returns a list of the rules before each row. The last entry will be the rule(s) following the last row.

`find_rule(table, row)` returns a [LaTeX2range](#) for the rule before row, not including the final whitespace.

`rule(table, row)` returns the rule(s) before row.

**See Also**

Use `index_to_path()` to convert to a path.

**Examples**

```

latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
table <- prepare_table(table)
find_rules(table)

rules(table)

find_rule(table, 1)

rule(table, 1)

```

```
rule(table, 2) <- "\\midrule"  
table
```

---

tables

*Functions related to parsing LaTeX tables*

---

## Description

Functions related to parsing LaTeX tables

## Usage

```
is_tabular(item)  
  
find_tabular(items, start = 1)
```

## Arguments

item	An item from a <a href="#">LaTeX2</a> list object.
items	A <a href="#">LaTeX2</a> list object.
start	Where to start looking.

## Value

`is_tabular()` returns boolean indicating if this is a tabular-like environment.  
`find_tabular()` returns the index of the first tabular-like environment, or NA if none is found.

## Examples

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")  
parsed <- parseLatex(latex)  
is_tabular(parsed[[2]])
```

```
find_tabular(parsed)  
table <- parsed[[find_tabular(parsed)]]  
table
```

tests

*Test objects***Description**

Test objects

**Usage**`is_env(item, envtypes = NULL)``is_macro(item, macros = NULL)``is_block(item)``is_bracket(item, bracket)``is_whitespace(item)``is_text(item)``is_error(item)``is_itemlist(item)``is_placeholder(item)``is_char(item, char)``is_catcode(item, code)`**Arguments**

<code>item</code>	An object of class <a href="#">LaTeX2item</a> to test.
<code>envtypes</code>	Types of Latex environment to check for, e.g. "table".
<code>macros</code>	Which macros to match, e.g. "\\caption".
<code>bracket</code>	Which bracket are we looking for?
<code>char</code>	A character to match
<code>code</code>	A catcode to match

**Value**`is_env()` returns a boolean if the item matches.`is_macro()` returns a boolean indicating the match.`is_block()` returns a boolean indicating whether the `item` is a block wrapped in curly braces.



`is_bracket()` returns a boolean indicating that the `item` is a bracket of the specified type.  
`is_whitespace()` returns a boolean indicating if the `item` is a space, tab or newline.  
`is_text()` returns a boolean indicating if the `item` is text.  
`is_error()` returns a boolean indicating if the `item` is an error.  
`is_itemlist()` returns a boolean indicating if the `item` is an [ITEMLIST](#) item.  
`is_placeholder()` returns a boolean indicating if the `item` is a [PLACEHOLDER](#) item.  
`is_char()` returns a boolean indicating if the `item` is a [SPECIAL](#) matching char.  
`is_catcode()` returns a boolean indicating if the `item` is a [SPECIAL](#) with the given catcode.

### Examples

```
is_bracket(parseLatex("[ ]")[[1]], "[")
```

---

 Utilities

---

*Miscellaneous utilities*


---

### Description

Miscellaneous utilities

### Usage

```

drop_items(items, which)

select_items(items, which)

drop_whitespace(items)

trim_whitespace(items)

include_whitespace(items, which)

split_chars(item, split = "")

new_block(...)

new_env(name, ...)
  
```

### Arguments

<code>items</code>	A <a href="#">LaTeX2</a> object or list of items, or a <a href="#">LaTeX2item</a> which is a list.
<code>which</code>	A <a href="#">LaTeX2range</a> object describing which items to operate on, or a vector of indices into <code>items</code> .
<code>item</code>	A non-list <a href="#">LaTeX2item</a> .

split	Where to split the characters.
...	Items to be passed to latex2().
name	The desired environment name.

**Value**

drop\_items() returns the list of items with specific items removed.

select\_items() returns the list of subsetted items.

drop\_whitespace() returns the items with whitespace (blanks, tabs, newlines) removed.

trim\_whitespace() returns the items with leading and trailing whitespace (blanks, tabs, newlines) removed.

include\_whitespace() returns which with following whitespace (blanks, tabs, newlines) included.

split\_chars() returns a [LaTeX2](#) list containing the result of calling [strsplit](#) on the text of the item.

new\_block() returns a BLOCK item containing the items.

new\_env() returns an environment item containing the other items.

**Note**

drop\_whitespace() will drop the whitespace that separates text items, so deparsing will merge them into a single item.

**See Also**

drop\_whitespace() does not act recursively; use [reduce\\_whitespace](#) for that.

**Examples**

```
parsed <- parseLatex("Hello")
unclass(parsed)
unclass(split_chars(parsed[[1]]))
new_block(parseLatex("abc"))
new_env("itemize", parseLatex("\\item An item"))
```

---

vector\_to\_latex2

*Convert vector to items*


---

**Description**

Convert vector to items

**Usage**

```
vector_to_latex2(x)
```

**Arguments**

x                    A list or vector to convert.

**Value**

A [LaTeX2](#) object containing the entries of x concatenated.

**Examples**

```
print(vector_to_latex2(1:3), tags = TRUE)
```

---

vector_to_row	<i>Convert vector to table row and back</i>
---------------	---

---

**Description**

Convert vector to table row and back

**Usage**

```
vector_to_row(cells, asis = FALSE, linebreak = TRUE)
```

```
row_to_vector(row, asis = FALSE, deparse = TRUE)
```

**Arguments**

cells                A list or vector of cell contents.  
 asis                 If FALSE, add or remove blanks around cell contents.  
 linebreak           If TRUE, add a line break marker.  
 row                  A row from a table  
 deparse             Should the result be deparsed?

**Value**

vector\_to\_row returns a [LaTeX2](#) object which could be a row in a tabular object.

row\_to\_vector returns a character vector of the deparsed contents of the row, or if deparse is FALSE, a list of the contents.

**Examples**

```
vector_to_row(1:3)
row_to_vector("1 & 2 & content \\\\")
row_to_vector("1 & 2 & content \\\\", deparse = FALSE)
```

# Index

- \* **datasets**
  - defaultCatcodes, 3
- ACTIVE (defaultCatcodes), 3
- ALIGN (defaultCatcodes), 3
- as\_LaTeX2, 2
  
- brace\_options (options), 15
- brace\_options<- (options), 15
- bracket\_options (options), 15
- bracket\_options<- (options), 15
  
- catcode (names), 14
- columnOption (tableOption), 28
- columnOption<- (tableOption), 28
- columnOptions (tableOption), 28
- columnOptions<- (tableOption), 28
- COMMENT (defaultCatcodes), 3
  
- defaultCatcodes, 3, 17
- deparseLatex, 4, 22
- drop\_caption (find\_caption), 6
- drop\_items (Utilities), 33
- drop\_whitespace (Utilities), 33
  
- envName (names), 14
- envName<- (names), 14
- ESCAPE (defaultCatcodes), 3
  
- find\_block (finders), 4
- find\_brace\_options (options), 15
- find\_bracket\_options (options), 15
- find\_caption, 6
- find\_catcode (finders), 4
- find\_char (finders), 4
- find\_columnOptions (tableOption), 28
- find\_env (finders), 4
- find\_general (finders), 4
- find\_general(), 15
- find\_macro (finders), 4
- find\_pattern, 7
  
- find\_posOption (tableOption), 28
- find\_rule (tableRule), 29
- find\_rules (tableRule), 29
- find\_sequence, 8
- find\_tableCell (tableCell), 26
- find\_tableContent, 9
- find\_tableRow, 10
- find\_tabular (tables), 31
- find\_tags (finders), 4
- find\_whitespace (finders), 4
- find\_widthOption (tableOption), 28
- finders, 4
- flatten\_itemlists (itemlist), 12
- flatten\_itemlists(), 21
  
- get\_container (path\_to), 18
- get\_contents, 11
- get\_item (path\_to), 18
- get\_items (path\_to), 18
- get\_leftovers, 12
- get\_range (set\_range), 23
- get\_ranges (path\_to\_index), 20
- get\_which (path\_to), 18
  
- IGNORE (defaultCatcodes), 3
- include\_whitespace (Utilities), 33
- index\_to\_path (path\_to\_index), 20
- index\_to\_path(), 6, 30
- insert\_values (path\_to), 18
- INVALID (defaultCatcodes), 3
- is\_block (tests), 32
- is\_bracket (tests), 32
- is\_catcode (tests), 32
- is\_char (tests), 32
- is\_env (tests), 32
- is\_error (tests), 32
- is\_itemlist (tests), 32
- is\_macro (tests), 32
- is\_placeholder (tests), 32
- is\_tabular (tables), 31

- is\_text (tests), 32
- is\_whitespace (tests), 32
- ITEMLIST, 21, 33
- ITEMLIST (itemlist), 12
- itemlist, 12
- items\_are\_equal (find\_sequence), 8
  
- LaTeX2, 3, 6–11, 13–16, 19, 20, 22–25, 27–31, 33–35
- LaTeX2 (parseLatex\_fn), 16
- latex2 (as\_LaTeX2), 2
- LaTeX2item, 6, 8, 13, 14, 19–21, 23, 24, 32, 33
- LaTeX2item (parseLatex\_fn), 16
- LaTeX2range, 6–8, 10, 13, 16, 20, 24, 27, 29, 30, 33
- latexTag (names), 14
- LBRACE (defaultCatcodes), 3
- LETTER (defaultCatcodes), 3
  
- macroName (names), 14
- MATH (defaultCatcodes), 3
  
- names, 14
- new\_block (Utilities), 33
- new\_env (Utilities), 33
- new\_itemlist (itemlist), 12
- NEWLINE (defaultCatcodes), 3
  
- options, 15
- OTHER (defaultCatcodes), 3
  
- PARAM (defaultCatcodes), 3
- parseLatex, 3
- parseLatex (parseLatex\_fn), 16
- parseLatex-package (parseLatex\_pkg), 18
- parseLatex\_fn, 16
- parseLatex\_pkg, 18
- path\_to, 18
- path\_to\_caption (find\_caption), 6
- path\_to\_index, 20
- path\_to\_index(), 6
- paths\_to\_ranges (path\_to\_index), 20
- paths\_to\_ranges(), 20
- PLACEHOLDER, 33
- PLACEHOLDER (itemlist), 12
- placeholder (itemlist), 12
- posOption (tableOption), 28
- posOption<- (tableOption), 28
- prepare\_row (prepare\_table), 21
- prepare\_table, 13, 21
- prepare\_table(), 8, 9
- print.LaTeX2 (print.LaTeX2item), 22
- print.LaTeX2item, 22
- print.LaTeX2range (LaTeX2range), 13
  
- RBRACE (defaultCatcodes), 3
- reduce\_whitespace, 22, 34
- rmSrcrefs, 23
- row\_to\_vector (vector\_to\_row), 35
- rule (tableRule), 29
- rule<- (tableRule), 29
- rules (tableRule), 29
  
- select\_items (Utilities), 33
- set\_contents (get\_contents), 11
- set\_item (path\_to), 18
- set\_range, 23
- show\_itemlists (itemlist), 12
- showErrors, 17, 24
- SPACE (defaultCatcodes), 3
- split\_chars (Utilities), 33
- split\_latex (splitting), 25
- split\_list (splitting), 25
- splitting, 25
- strsplit, 34
- SUB (defaultCatcodes), 3
- SUPER (defaultCatcodes), 3
  
- tablecalcs, 26
- tableCell, 26
- tableCell<- (tableCell), 26
- tableContent (find\_tableContent), 9
- tableContent<- (find\_tableContent), 9
- tableDim (tablecalcs), 26
- tableNcol (tablecalcs), 26
- tableNrow (tablecalcs), 26
- tableOption, 28
- tableRow (find\_tableRow), 10
- tableRow<- (find\_tableRow), 10
- tableRule, 29
- tables, 31
- tests, 32
- trim\_whitespace (Utilities), 33
  
- Utilities, 33
  
- vector\_to\_latex2, 34
- vector\_to\_row, 35

`widthOption(tableOption)`, 28  
`widthOption<- (tableOption)`, 28