Package ‘parseLatex’

February 20, 2025

Type Package

Title Parse 'LaTeX' Code

Version 0.3.0

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

Imports utils

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2025-02-20 16:30:02 UTC

Contents

`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` *The default "catcodes" used by parseLatex.*

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.

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))
```

`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)

find_macro(items, macros)

find_catcode(items, codes)

find_tags(items, tags)

find_char(items, char)
```

Arguments

`items`	A list of latex items.
`envtypes`	Which types of environment to look for.
`macros`	Which types of macros to look for.
`codes`	Which codes to look for.
`tags`	Which tags to look for.
`char`	Which character to look for.

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_pattern` *Find a pattern in deparsed items*

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

`items`	A list of latex items.
`pattern`	Pattern to use in `grepl()`.
`...`	Additional parameters to pass to `grepl`.
`all`	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 *Find a code sequence*

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.

Value

`find_sequence()` returns a path or list of paths 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.
`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.

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)

tableRow(table, row)

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

Arguments

`table`	A tabular-like environment to work with.
`row`	row in the table (1 is top row), including rows of labels.
`asis`	Should a linebreak and newline be added after 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.

Details

Unless `asis = TRUE`, `tableContent(table) <- value` will add "\\" 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.

Value

`find_tableRow()` returns the indices 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, 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.

The original `item` with the contents replaced by `value`.

Examples

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

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

`get_leftovers` *Retrieve source from beyond the end of the document.*

Description

Retrieve source from beyond the end of the document.

Usage

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

Arguments

`text`	Character vector holding source.
`items`	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)
```

`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)

macroName(item)
```

Arguments

`item`	A single latex item.

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 = 1, start = 1)

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

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

find_brace_options(items, which = 1, start = 1)

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]]`.
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.

Value

`find_bracket_options` returns indices into `items` of the options (including the brackets).

`bracket_options` returns a LaTeX2 object containing the specified options.

`find_brace_options` returns the index of 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
)

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

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

Arguments

`text`	A character vector containing LaTeX source code.
`verbose`	If `TRUE`, print debug error messages.
`verbatim`	A character vector containing the names of LaTeX 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.
`catcodes`	A list or dataframe holding LaTeX "catcodes", such as defaultCatcodes.
`recover`	If `TRUE`, convert errors to warnings and continue parsing. See Details below.
`x`	Object to work on.
`...`	Extra parameters to pass to `deparseLatex`.
`tags`	Whether to display LaTeX2 tags.

Details

Some versions of LaTeX such as `pdflatex` 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 `iswalpha(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"`.

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, is_fn, ..., all = FALSE)

get_item(items, path)

set_item(items, path, value)

get_container(items, path)

get_which(path)
```

Arguments

`items`	A list of latex items.
`is_fn`	Which test function to use.
`...`	Additional parameters to pass to `is_fn`.
`all`	Find all matching, or the first?
`path`	Integer vector of subitems
`value`	A LaTeX2item to set as a value.

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.

`set_item()` replaces the item at the given path, 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, is_fn = is_env,
                        envtypes = "tabular")
get_item(parsed, path)
```

`reduce_whitespace` *Remove excess whitespace recursively*

Description

Remove excess whitespace recursively

Usage

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

Arguments

`items`	A LaTeX2 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)
```

`set_range` *Set items in a LaTeX2 object*

Description

Set items in a LaTeX2 object

Usage

```
set_range(items, range, values)
```

Arguments

`items`	A LaTeX2 object or other list of LaTeX2item objects.
`range`	A LaTeX2range object.
`values`	An object that can be coerced to a LaTeX2 object or (if `range$range` is `NULL`) a LaTeX2item.

Value

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

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, 8)
set_range(parsed, range, "The 8th item")
```

`tablecalcs` *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 LaTeX2 object, or a character string that will be converted to one.

Details

`find_tableCell()` returns `NA` 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 the indices of the entries corresponding to the content of the cell (row, col) of the table.

`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 <- parsed[[find_tabular(parsed)]]
find_tableCell(table, 1, 2)

tableCell(table, 1, 2)

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

`tableOption` *Functions related to table options.*

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)

columnOptions(table, asis = FALSE) <- 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 LaTeX2 object.

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 the index of the block corresponding to the column spec.

`columnOptions()` returns a LaTeX2 object containing the "column" options.

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)

columnOptions(table) <- "lrr"
```

```
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)

find_rule(table, row)

rule(table, row)

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

Arguments

`table`	A tabular-like environment to work with.
`row`	The rules will precede the contents of this row. The rule after the final row uses `row = tableNrow(table) + 1`.
`asis`	Should a newline be added after the value? If `asis = TRUE`, it will not be.
`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.

Value

`find_rules()` returns a list of the indices of rules before each row, including the whitespace following each one.

`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 the indices of the rule(s) before `row`.

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

Examples

```
latex <- kableExtra::kbl(mtcars[1:2, 1:2], format = "latex")
parsed <- parseLatex(latex)
table <- parsed[[find_tabular(parsed)]]
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 LaTeX2 list object.
items	A LaTeX2 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)
```

Arguments

`item`	An object of class LaTeX2item to test.
`envtypes`	Types of Latex environment to check for, e.g. `"table"`.
`macros`	Which macros to match, e.g. `"\\\\caption"`.
`bracket`	Which bracket are we looking for?

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.

Examples

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

`Utilities` *Miscellaneous utilities*

Description

Miscellaneous utilities

Usage

```
drop_items(items, which)

select_items(items, which)

drop_whitespace(items)

include_whitespace(items, which)

split_list(items, splits)

split_latex(items, splits)

new_block(items)
```

Arguments

`items`	A LaTeX2 object or list of items.
`which`	Which items to operate on.
`splits`	Which items divide the parts?

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.

`include_whitespace()` returns `which` with following whitespace (blanks, tabs, newlines) included.

`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 a LaTeX2 object.

A BLOCK item containing the 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

```
new_block(parseLatex("abc"))
```

`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` *Convert vector to table row*

Description

Convert vector to table row

Usage

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

Arguments

`cells`	A list or vector of cell contents.
`asis`	If `FALSE`, add blanks around cell contents.
`linebreak`	If `TRUE`, add a line break marker.

Value

A LaTeX2 object which could be a row in a tabular object.

Examples

```
vector_to_row(1:3)
```

Index