Package ‘mathml’

May 22, 2026

Type Package

Title Translate R Expressions to 'MathML' and 'LaTeX'/'MathJax'

Version 1.8

Date 2026-05-22

Maintainer Matthias Gondan <Matthias.Gondan-Rochon@uibk.ac.at>

Description Translate R expressions to 'MathML' or 'MathJax'/'LaTeX' so that they can be rendered in R markdown documents and shiny apps. This package depends on R package 'rolog', which requires an installation of the 'SWI'-'Prolog' runtime either from 'swi-prolog.org' or from R package 'rswipl'.

License FreeBSD

Depends R (>= 4.3), rolog (>= 0.9.14), knitr

Imports xfun (>= 0.49)

Encoding UTF-8

URL https://github.com/mgondan/mathml

BugReports https://github.com/mgondan/mathml/issues

Suggests rmarkdown, testthat, rswipl

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Matthias Gondan [aut, cre, cph] (University of Innsbruck), Irene Alfarone [aut] (University of Innsbruck), European Commission [fnd] (Erasmus+ Programme, 2019-1-EE01-KA203-051708)

Repository CRAN

Date/Publication 2026-05-22 09:40:08 UTC

Contents

add . . . 3
add_left . . . 3
add_right . . . 4
cal . . . 4
canonical . . . 5
decorations . . . 5
denote . . . 6
dfrac . . . 7
dot . . . 7
fname . . . 8
fontstyles . . . 8
frac . . . 9
hook . . . 9
instead . . . 10
math . . . 11
mathjax . . . 11
mathml . . . 12
mathml_preproc . . . 13
mathout . . . 14
name . . . 15
omit . . . 15
omit_left . . . 16
omit_right . . . 16
prod_over . . . 17
sum_over . . . 17
%.% . . . 18
%dbldown% . . . 19
%dblup% . . . 19
%down% . . . 20
%==% . . . 20
%=>% . . . 21
%=~% . . . 21
%->% . . . 22
%<=% . . . 22
%<=>% . . . 23
%+-% . . . 23
%prop% . . . 24
%<-% . . . 24
%<->% . . . 25
%~~% . . . 25
%up% . . . 26

Index **27**

add *add*

Description

This is a function that allows the user to highlight the mistakes, in particular an extra element in a list

Usage

```
add(expr)
```

Arguments

expr	expression

Value

expr , e.g., highlights a + b from a + b

add_left *add_left*

Description

This is a function that allows the user to highlight the mistakes, in particular the redundancies in the left-hand side of the expression.

Usage

```
add_left(expr)
```

Arguments

expr	expression

Value

expr e.g., highlights a + from a + b

add_right *add_right*

Description

This is a function that allows the user to highlight the mistakes, in particular the redundancies in the right-hand side of the expression.

Usage

```
add_right(expr)
```

Arguments

expr	expression

Value

expr , e.g., highlights + b from a + b

cal *Calligraphic font*

Description

Calligraphic font

Usage

```
cal(x)
```

Arguments

x	an R symbol. This function is used to render the content in calligraphic font in MathJax. In MathML, script font is used.

Value

The function cal is a wrapper for the identity function.

See Also

identity()

Examples

```
mathjax(quote(K %in% cal(K)))
```

canonical *Canonicalize an R call: Reorder the function arguments*

Description

Canonicalize an R call: Reorder the function arguments

Usage

```
canonical(term = quote(`%in%`(table = Table, x = X)), drop = TRUE)
```

Arguments

term	an R call.
drop	whether to drop the argument names or not

Value

The R function, with arguments rearranged

Examples

```
canonical(term=quote(`%in%`(table=Table, x=X)))
```

decorations *Identity functions for different decorations*

Description

Identity functions for different decorations

Usage

```
roof(x)

boxed(x)

cancel(x)

phantom(x)

prime(x)

tilde(x)
```

```
over(x)

under(x)

underover(x)

hyph(x)

color(x)
```

Arguments

x	the expression to render

Value

x

Examples

```
roof(1) + mean(2) + boxed(3) + cancel(4) + phantom(5) + prime(6) + tilde(7)

mathml(quote(roof(b) + mean(X) + boxed(3) + cancel(4) + phantom(5)))
```

`denote` *denote This is a function that allows the user to insert abbreviations in the formula, explain them and make the needed computations*

Description

denote This is a function that allows the user to insert abbreviations in the formula, explain them and make the needed computations

Usage

```
denote(abbr, expr, info)
```

Arguments

abbr	Abbreviation used in the text to refer to the calculation, for example 's_p' for the pooled variance.
expr	Expression: calculations to be made in order to obtain the value to which the abbreviation refers to.
info	Information: Explanation of the formula used to provide the value of the abbreviation. e.g. 'the pooled variance'

Value

expr e.g., x denotes a^2 + b

dfrac *Division displayed as large fraction*

Description

Division displayed as large fraction

Usage

```
dfrac(e1, e2)
```

Arguments

e1	numerator
e2	denominator

Value

e1 / e2

See Also

frac(), over()

dot *Multiplication*

Description

Multiplication

Usage

```
dot(e1, e2)

nodot(e1, e2)

times(e1, e2)
```

Arguments

e1	numerator
e2	denominator

Value

e1 * e2

fname *Return function body*

Description

Return function body

Usage

```
fname(fname, body)
```

Arguments

fname	not clear
body	not clear

Value

body

fontstyles *Identity functions for different font styles*

Description

Identity functions for different font styles

Usage

```
plain(x)

italic(x)

bold(x)
```

Arguments

x	the expression to render

Value

x

Examples

```
plain(1) + bold(2) + italic(3)

mathml(term=quote(plain(abc) + bold(def) + italic(ghi)))
```

frac *Division displayed as fraction*

Description

Division displayed as fraction

Usage

```
frac(e1, e2)
```

Arguments

e1	numerator
e2	denominator

Value

e1 / e2

hook *Hook for custom symbols*

Description

hook(term, display) hook_fn(fn) unhook(term) hooked(term)

Usage

```
hook(term, display = NULL, quote = TRUE, as.rolog = TRUE)

unhook(term, quote = TRUE, as.rolog = TRUE)

hooked(term)

hook_fn(fn)
```

Arguments

term	an R call or symbol/number. This is the expression to replace.
display	an R call or symbol/number. This is shown instead of *term*.
quote	(default is TRUE) indicates that *term* and *display* should be quoted.
as.rolog	(default is TRUE) indicates that simplified quasi-quotation is to be used.
fn	a custom function. The name of *fn* is replaced by its function body.

Value

hook and unhook return TRUE on success. hooked returns the hooked expression or FALSE on failure.

Examples

```
hook(t0, subscript(t, 0))
hooked(quote(t0))
mathml(quote(t0))
hook(term=quote(t0), display=quote(superscript(t, 0)), quote=FALSE)
mathml(quote(t0))
unhook(t0)
mathml(quote(t0))
square <- function(x) {x^2} ; hook_fn(square)
```

`instead` *instead*

Description

This is a function that allows the user to highlight the mistakes, in particular adds a curly bracket under the wrong term and it provides the correct solutions.

Usage

```
instead(inst, of)
```

Arguments

`inst`	the wrong term
`of`	the correct term

Value

inst

Examples

```
1 + instead(2, 3)

mathml(term=quote(1 + instead(2, 3)))
```

math *Adds the class "math" to the object for knitr output via* `mathout()`

Description

Adds the class "math" to the object for knitr output via `mathout()`

Usage

```
math(term, flags = NULL)
```

Arguments

`term`	an R call or symbol/number. This function translates *term* into a LaTeX/MathJax string.
`flags`	(default NULL) list of flags that control the translation

Value

term with additional class "math" and *flags* as attributes.

See Also

`mathml()`, `mathjax()`, `mathout()`

Examples

```
math(term=quote((a + b)^2L == a^2L + 2L*a*b + b^2L))
```

mathjax *Mathjax output*

Description

Mathjax output

Usage

```
mathjax(
  term = quote((a + b)^2L == a^2L + 2L * a * b + b^2L),
  flags = NULL,
  env = globalenv()
)
```

Arguments

term	an R call or symbol/number. This function translates *term* into a LaTeX/MathJax string.
flags	(default NULL) list of flags that control the translation
env	(default globalenv()) The R environment in which r_eval is being executed (see vignette for details, "Ringing back to R").

Details

In some functions, the Prolog code may ring back R, for example, to find the names of function arguments. For example (see vignette), when rendering the call `integrate(g, lower=0L, upper=Inf)` as Int_0^Inf g(x) dx, Prolog needs to know that the function g is a function of x. The Prolog rule then searches for the formalArgs of g in the environment *env*.

Value

A string with the MathJax representation of *term*.

See Also

`mathml()`

Examples

```
mathjax(term=quote((a + b)^2L == a^2L + 2L*a*b + b^2L))
```

---

mathml *MathML output*

Description

MathML output

Usage

```
mathml(
  term = quote((a + b)^2L == a^2L + 2L * a * b + b^2L),
  flags = NULL,
  env = globalenv()
)
```

Arguments

term	an R call or symbol/number. This function translates *term* into a MathML string.
flags	(default NULL) list of flags that control the translation. This includes "context" settings such as error("ignore"), or a default number of decimal places for numeric output.
env	(default globalenv()) The R environment in which r_eval is being executed.

Details

In some functions, the Prolog code may ring back R, for example, to find the names of function arguments. For example (see vignette), when rendering the call `integrate(g, lower=0L, upper=Inf)` as Int_0^Inf g(x) dx, Prolog needs to know that the function g is a function of x. The Prolog rule then searches for the formalArgs of g in the environment *env*.

Value

A string with the MathML representation of *term*.

See Also

`mathjax()`

Examples

```
mathml(term=quote((a + b)^2L == a^2L + 2L*a*b + b^2L))
mathml(term=3.14159265, flags=list(round=3L))
mathml(term=3.14159265, flags=list(quote(round(3L))))
```

---

mathml_preproc *Map R operators to their respective Prolog counterparts*

Description

Map R operators to their respective Prolog counterparts

Usage

```
mathml_preproc(query = quote(5%%2))
```

Arguments

`query`	an R call or symbol/number. This function translates components of *query* into their respective counterparts from Prolog

Value

The translated query

See Also

`mathjax()`, `mathml()`

Examples

```
mathml_preproc(quote(5 %% 2))
```

mathout *MathML or MathJax output, depending on the knitr context*

Description

MathML or MathJax output, depending on the knitr context

Usage

```
mathout(term, flags = NULL, env = parent.frame())

inline(term, flags = NULL, env = parent.frame())
```

Arguments

term	an R call or symbol/number. This function translates *term* into a LaTeX/MathJax string.
flags	(default NULL) list of flags that control the translation
env	(default parent.frame()) The R environment in which r_eval is being executed (see vignette for details, "Ringing back to R").

Details

This function checks knitr::is_html_output() and knitr::is_html_output() and invokes the respective function mathml() or mathjax(). Outside of knitr context, MathML is returned, and a warning is given.

Value

A string with the MathML or MathJax representation of *term*.

See Also

mathml(), mathjax(), inline()

Examples

```
mathout(term=quote((a + b)^2L == a^2L + 2L*a*b + b^2L))

inline(term=quote((a + b)^2L == a^2L + 2L*a*b + b^2L))
```

name *Add a name attribute to an element (most often, an R function)*

Description

Add a name attribute to an element (most often, an R function)

Usage

```
name(x, name)
```

Arguments

x	an R object, e.g., an R function
name	the name of the object/function

Value

The object with the name attribute

Examples

```
f <- function(x) {sin(x)}
mathjax(call("integrate", name(f, "sin"), 0L, 2L*pi))
```

omit *omit*

Description

This is a function that allows the user to highlight the mistakes, in particular the omission of an element from a list.

Usage

```
omit(expr)
```

Arguments

expr	expression

Value

NULL e.g., remove a + b from a + b

`omit_left` *omit_left This is a function that allows the user to highlight the mistakes, in particular the omissions in the left-hand side of the expression*

Description

omit_left This is a function that allows the user to highlight the mistakes, in particular the omissions in the left-hand side of the expression

Usage

```
omit_left(expr)
```

Arguments

`expr`	The expression, e.g. a + b

Value

substitute(expr)[[3]], e.g., b from a + b

`omit_right` *omit_right This is a function that allows the user to highlight the mistakes, in particular the omissions in the right-hand side of the expression*

Description

omit_right This is a function that allows the user to highlight the mistakes, in particular the omissions in the right-hand side of the expression

Usage

```
omit_right(expr)
```

Arguments

`expr`	expression

Value

substitute(expr)[[2]], e.g., a from a + b

prod_over *product over a range. On the R side, this function just returns the product of the first argument, but allows for decorations.*

Description

product over a range. On the R side, this function just returns the product of the first argument, but allows for decorations.

Usage

```
prod_over(x, from, to)
```

Arguments

x	the object to be multiplied
from	decoration for prod_from^to x_i
to	decoration for prod_from^to x_i

Value

The function returns prod(x)

See Also

prod(), sum_over()

Examples

```
mathjax(quote(prod_over(x[i], i=1L, N)))
```

sum_over *sum over a range. On the R side, this function just returns the first argument, but allows for decorations.*

Description

sum over a range. On the R side, this function just returns the first argument, but allows for decorations.

Usage

```
sum_over(x, from, to)
```

Arguments

x	the object to be summed
from	decoration for sum_from^to x_i
to	decoration for sum_from^to x_i

Value

The function returns sum(x)

See Also

sum(), prod_over()

Examples

```
mathjax(quote(sum_over(x[i], i=1L, N)))
```

%.% *Product x * y, shown as x dot y*

Description

Product x * y, shown as x dot y

Usage

```
x %.% y
```

Arguments

x	first factor
y	second factor

Value

x * y

%dbldown% *Down double arrow, displayed as x dArr y*

Description

Down double arrow, displayed as x dArr y

Usage

```
x %dbldown% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a downward double arrow

%dblup% *Up double arrow, displayed as x uArr y*

Description

Up double arrow, displayed as x uArr y

Usage

```
x %dblup% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a upward double arrow

`%down%` *Down arrow, presented as x downarrow y*

Description

Down arrow, presented as x downarrow y

Usage

```
x %down% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a downward arrow

`%==%` *Equivalence, shown as x == y*

Description

Equivalence, shown as x == y

Usage

```
x %==% y
```

Arguments

x	first argument
y	second argument

Value

x == y

`%=>%` *Left double arrow, displayed as x <= y*

Description

Left double arrow, displayed as x <= y

Usage

```
x %=>% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a left double arrow

`%=~%` *Congruence, shown as x =~ y*

Description

Congruence, shown as x =~ y

Usage

```
x %=~% y
```

Arguments

x	first argument
y	second argument

Value

x == y, e.g., a cong b

`%->%` *Right arrow, presented as x -> y*

Description

Right arrow, presented as x -> y

Usage

```
x %->% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a right arrow

`%<=%` *Right double arrow, displayed as x => y*

Description

Right double arrow, displayed as x => y

Usage

```
x %<=% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a right double arrow

%<=>% *If and only if condition, displayed as x <=> y*

Description

If and only if condition, displayed as x <=> y

Usage

```
x %<=>% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a double arrow double-sided

%+-% *Plus Minus, it shows x and calculates x +- y*

Description

Plus Minus, it shows x and calculates x +- y

Usage

```
x %+-% y
```

Arguments

x	first term
y	second term

Value

c(x - y, x + y) x plus min y

%prop% *Proportional, shown as x o< y*

Description

Proportional, shown as x o< y

Usage

```
x %prop% y
```

Arguments

x	first argument
y	second argument

Value

NA

%<-% *Left arrow, presented as x <- y*

Description

Left arrow, presented as x <- y

Usage

```
x %<-% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a left arrow

%<->% *Double sided arrow, presented as x <-> y*

Description

Double sided arrow, presented as x <-> y

Usage

```
x %<->% y
```

Arguments

x	first element
y	second element

Value

NA, it produces a double sided arrow

%~~% *Approximate equality, shown as x ~~ y*

Description

Approximate equality, shown as x ~~ y

Usage

```
x %~~% y
```

Arguments

x	first argument
y	second argument

Value

The result of isTRUE(all.equal(x, y))

%up% *Up arrow, presented as x up y*

Description

Up arrow, presented as x up y

Usage

```
x %up% y
```

Arguments

x	first element
y	second element

Value

NA, it produces an upward arrow

Index

%+-%, 23
%->%, 22
%.%, 18
%<->%, 25
%<-%, 24
%<=>%, 23
%<=%, 22
%==%, 20
%=>%, 21
%=~%, 21
%~~%, 25
%dbldown%, 19
%dblup%, 19
%down%, 20
%prop%, 24
%up%, 26

add, 3
add_left, 3
add_right, 4

bold *(*fontstyles*)*, 8
boxed *(*decorations*)*, 5

cal, 4
cancel *(*decorations*)*, 5
canonical, 5
color *(*decorations*)*, 5

decorations, 5
denote, 6
dfrac, 7
dot, 7

fname, 8
fontstyles, 8
frac, 9
frac(), *7*

hook, 9
hook_fn *(*hook*)*, 9
hooked *(*hook*)*, 9
hyph *(*decorations*)*, 5

identity(), *4*
inline *(*mathout*)*, 14
inline(), *14*
instead, 10
italic *(*fontstyles*)*, 8

math, 11
mathjax, 11
mathjax(), *11*, *13*, *14*
mathml, 12
mathml(), *11–14*
mathml_preproc, 13
mathout, 14
mathout(), *11*

name, 15
nodot *(*dot*)*, 7

omit, 15
omit_left, 16
omit_right, 16
over *(*decorations*)*, 5
over(), *7*

phantom *(*decorations*)*, 5
plain *(*fontstyles*)*, 8
prime *(*decorations*)*, 5
prod(), *17*
prod_over, 17
prod_over(), *18*

roof *(*decorations*)*, 5

sum(), *18*
sum_over, 17
sum_over(), *17*

tilde *(*decorations*)*, 5

times *(*dot*)*, 7

under *(*decorations*)*, 5
underover *(*decorations*)*, 5
unhook *(*hook*)*, 9