Package: gridmicrotex (via r-universe)

October 5, 2026

Title Native 'LaTeX' Math Rendering for Grid Graphics

Version 0.2.0

Description Renders 'LaTeX' math equations as native R grid graphics objects (grobs) using the 'MicroTeX' 'C++' library as the layout engine. Produces resolution-independent vector output that works on any R graphics device, with no external 'LaTeX' installation required. Markdown labels and block documents that mix prose formatting with math are also rendered, for use with both 'grid' and 'ggplot2'.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements C++17, FreeType (>= 2.9), pkg-config, FriBidi (optional)

Depends R (>= 4.2.0)

LinkingTo Rcpp, systemfonts

Imports commonmark, grDevices, grid, Rcpp, systemfonts (>= 1.2.0), tools, utils, xml2

Suggests ggplot2 (>= 4.0.0), grImport2, jpeg, knitr, png, ragg, rmarkdown, rsvg, S7, svglite, testthat (>= 3.0.0), textshaping, vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/adayim/gridmicrotex, https://adayim.github.io/gridmicrotex/

BugReports https://github.com/adayim/gridmicrotex/issues

RoxygenNote 7.3.3

NeedsCompilation yes

Author Alim Dayim [aut, cre] (ORCID: <https://orcid.org/0000-0001-9998-7463>), Nano Michael [cph] (Author of included 'MicroTeX' library), Bundled math font

authors [cph] (See inst/COPYRIGHTS for the full list of authors of the bundled math fonts.)

Maintainer Alim Dayim <ad938@cam.ac.uk>

Config/pak/sysreqs libfreetype6-dev libfribidi-dev pkg-config

Repository https://cran.r-universe.dev

Date/Publication 2026-10-05 01:03:43 UTC

RemoteUrl https://github.com/cran/gridmicrotex

RemoteRef HEAD

RemoteSha f5a1e534f56828b5e60cc54e83a1ee2061345093

Contents

available_fonts . . . 2
available_highlighters . . . 3
define_macro . . . 4
element_latex . . . 5
element_markdown . . . 6
geom_latex . . . 8
geom_markdown . . . 10
grobMark . . . 13
latex_cache_limit . . . 14
latex_dims . . . 15
latex_grob . . . 16
latex_options . . . 19
latex_tree . . . 21
latex_wrap . . . 22
load_font . . . 23
markdown_box_grob . . . 24
markdown_grob . . . 26
markdown_style . . . 28
md_style . . . 29
register_highlighter . . . 30

Index **32**

`available_fonts` *List the fonts that have been loaded*

Description

One row per font that can be named: the bundled math fonts, and every font given to `load_font()` or to a font option of `latex_options()`. The math fonts, which `math_font` takes, are the rows with `math` `TRUE`.

Usage

```
available_fonts(system = FALSE)
```

Arguments

system	If TRUE, also list the installed font families, which work by name without loading. The first call scans the system's fonts, which takes a few seconds.

Value

A data frame with one row per font: name; math (it has a math table; NA for an installed family that is not loaded); mono (monospaced); bold and italic (a face of its own is registered, so \textbf and \textit draw its real design); weight of the regular face; file of the regular face; and source, one of "bundled", "file" and "system".

Font pairing

For a consistent look, pair a math font with a matching fontfamily in gp:

Math font	Style	Suggested text font
Lete Sans Math ("lete", default)	Sans-serif	"sans"
STIX Two Math ("stix")	Serif	"serif"

See Also

load_font(), latex_options()

Examples

```
available_fonts()
```

---

available_highlighters

Syntax highlighting languages available

Description

Languages that can follow an opening code fence in markdown_box_grob(), including those added with register_highlighter(). Aliases such as py, sh, c++, yml, jl and tex also work. Other languages are shown as plain code.

Usage

```
available_highlighters()
```

Value

A sorted character vector.

See Also

register_highlighter

Examples

```
available_highlighters()
```

`define_macro` *Define a LaTeX shorthand for every label*

Description

`define_macro()` adds a macro without arguments, such as `\RR` for `\mathbb{R}`, that every later label can use. `list_macros()` shows them and `clear_macros()` removes them.

Usage

```
define_macro(name, definition)

clear_macros(name = NULL)

list_macros()
```

Arguments

`name`	Macro name, without the backslash. For `clear_macros()`, `NULL` (default) removes all macros.
`definition`	The LaTeX the macro stands for.

Details

A label can also define its own macros with `\newcommand`, `\def` and the like, including macros with arguments, but those last for that label only.

Value

`list_macros()` returns a named character vector of macros and their definitions. The others return `NULL`, invisibly.

See Also

latex_grob, latex_options

Examples

```
define_macro("RR", "\\mathbb{R}")
define_macro("eps", "\\varepsilon")
grid::grid.newpage()
grid.latex("\\forall \\eps > 0, \\eps \\in \\RR")
clear_macros()
```

`element_latex` *A ggplot2 theme element for LaTeX text*

Description

A theme element that renders text, such as an axis or plot title, as LaTeX. The `$` signs are optional. It takes the same settings as `ggplot2::element_text()` and inherits from the theme like it.

Usage

```
element_latex(
  math_font = "",
  fontsize = NULL,
  lineheight = 1.2,
  max_width = 0,
  input_mode = c("mixed", "math", "document"),
  render_mode = c("typeface", "path"),
  ...
)
```

Arguments

`math_font`	Math font: `"lete"` (Lete Sans Math, the default), `"stix"` (STIX Two Math), or one added with `load_font()`. See `available_fonts()`. A font that is not loaded, or has no math table, is an error.
`fontsize`	Font size in points. `NULL` (default) uses the theme's size.
`lineheight`	Line spacing (default 1.2).
`max_width`	Width in big points (1/72 inch) at which lines wrap. `0`, the default, does not wrap.
`input_mode`	How `tex` is read: • `"mixed"` (default): text, with math between `$...$` or `\(...\)`. A newline starts a new line. • `"math"`: everything is math; write text in `\text{}`. • `"document"`: a LaTeX document body, with paragraphs, numbered headings and displayed equations. Use it with `max_width`.

render_mode	"typeface" (default) draws glyphs as text, which can be selected in PDF and SVG output. It needs a device such as ragg, svglite or grDevices::cairo_pdf(), and falls back to "path" on others, such as pdf(). "path" draws glyphs as outlines, which works on every device.
...	Passed to ggplot2::element_text(), such as colour or hjust.

Value

A theme element of class element_latex, a kind of element_text.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(mtcars, aes(wt, mpg)) + geom_point() +
    labs(x = "$\\beta_1 \\cdot x + \\beta_0$") +
    theme(axis.title.x = element_latex())
}
```

---

element_markdown *A ggplot2 theme element for markdown text*

Description

A theme element that renders text, such as an axis or plot title, as markdown with $math$. It takes the same settings as ggplot2::element_text() and inherits from the theme like it.

Usage

```
element_markdown(
  math_font = "",
  fontsize = NULL,
  lineheight = 1.2,
  max_width = 0,
  render_mode = c("typeface", "path"),
  justify = FALSE,
  style = NA,
  width = NA,
  ...
)
```

Arguments

math_font	Math font, such as "stix".
fontsize	Font size in points. NULL (default) uses the theme's size.
lineheight	Line spacing (default 1.2).

max_width	Width in big points at which lines wrap. 0, the default, does not wrap.
render_mode	"typeface" (default) or "path".
justify	If TRUE, stretch wrapped lines to fill max_width.
style	A markdown_style(), CSS text, or a path to a .css file. NA (default) uses latex_options(markdown_style =).
width	Width at which the label wraps, as a grid::unit(). NA (default) does not wrap. unit(1, "npc") suits a plot title.
...	Passed to ggplot2::element_text(), such as colour or hjust.

Value

A theme element of class element_markdown, a kind of element_text.

Block labels

A label with a heading, a list, a table, a rule or several paragraphs is laid out as a small document, as in markdown_box_grob():

```
labs(title = "## Findings\n\n- slope $\\beta_1$\n- *p* < 0.001")
```

Its box is styled by the body rule, as in style = "body { background: grey95; padding: 8px }".

- Text wraps only if width is given.
- Axis tick labels and rotated labels are never laid out as blocks; a rotated one warns.
- math_font, render_mode and justify do not apply to blocks; set them with latex_options().

ggtext also has an element_markdown(). If both packages are attached, write gridmicrotex::element_markdown().

See Also

markdown_grob, element_latex

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(mtcars, aes(wt, mpg)) + geom_point() +
    labs(x = "**weight** in $10^3$ lbs") +
    theme(axis.title.x = gridmicrotex::element_markdown())
}
```

geom_latex *A ggplot2 geom for LaTeX math labels*

Description

Like ggplot2::geom_text(), with labels written in LaTeX. The $ signs are optional. annotate("latex", ...) adds a single label.

Usage

```
geom_latex(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  fontsize = 11,
  math_font = "",
  lineheight = 1.2,
  max_width = 0,
  input_mode = c("mixed", "math", "document"),
  render_mode = c("typeface", "path"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount.

- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Passed to `ggplot2::layer()`.

`fontsize` Font size in points, unless `size` is mapped.

`math_font` Math font, such as `"stix"`.

`lineheight` Line spacing (default 1.2).

`max_width` Width in big points at which lines wrap. `0`, the default, does not wrap.

`input_mode` How `tex` is read:

- `"mixed"` (default): text, with math between `$...$` or `\(...\)`. A newline starts a new line.
- `"math"`: everything is math; write text in `\text{}`.
- `"document"`: a LaTeX document body, with paragraphs, numbered headings and displayed equations. Use it with `max_width`.

`render_mode` `"typeface"` (default) draws glyphs as text, which can be selected in PDF and SVG output. It needs a device such as ragg, svglite or `grDevices::cairo_pdf()`, and falls back to `"path"` on others, such as `pdf()`. `"path"` draws glyphs as outlines, which works on every device.

`na.rm` If `FALSE` (default), missing values are removed with a warning; if `TRUE`, silently.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer.

Aesthetics

`geom_latex()` understands the following aesthetics (required aesthetics are in bold):

- `x`
- `y`
- `label`: LaTeX string
- `size`: font size in points (default: 11)
- `colour`: text colour (default: `"black"`)
- `angle`: rotation angle in degrees (default: 0)
- `hjust`: horizontal justification, 0-1 (default: 0.5)
- `vjust`: vertical justification, 0-1 (default: 0.5)
- `alpha`: transparency (default: 1)

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  df <- data.frame(
    x = 1:3, y = 1:3,
    eq = c("$x^2$", "$\\frac{a}{b}$", "$\\sum_{i=1}^n x_i$")
  )
  ggplot(df, aes(x, y, label = eq)) + geom_latex()

  # Use annotate() for single annotations (no legend, no data frame needed)
  ggplot(mtcars, aes(wt, mpg)) + geom_point() +
    annotate("latex", x = 4, y = 30,
             label = r"($\hat{y} = \beta_0 + \beta_1 x$)")
}
```

---

geom_markdown *A ggplot2 geom for markdown labels*

Description

Like `ggplot2::geom_text()`, with labels written in markdown: `**bold**`, `*italic*`, `` `code` ``, `~~strike~~` and `$math$`, as in `markdown_grob()`. `annotate("markdown", ...)` adds a single label.

Usage

```
geom_markdown(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
```

```
  ...,
  fontsize = 11,
  math_font = "",
  lineheight = 1.2,
  max_width = 0,
  render_mode = c("typeface", "path"),
  justify = FALSE,
  style = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

...	Passed to ggplot2::layer().
fontsize	Font size in points, unless size is mapped.
math_font	Math font, such as "stix".
lineheight	Line spacing (default 1.2).
max_width	Width in big points at which lines wrap. 0, the default, does not wrap.
render_mode	"typeface" (default) or "path"; see latex_grob().
justify	If TRUE, stretch wrapped lines to fill max_width.
style	A markdown_style(), CSS text, or a path to a .css file. NULL uses latex_options(markdown_style =). Only properties marked *inline* in md_style() apply.
na.rm	If FALSE (default), missing values are removed with a warning; if TRUE, silently.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer.

Aesthetics

geom_markdown() understands the following aesthetics (required aesthetics are in bold):

- x
- y
- label: markdown string
- size: font size in points (default: 11)
- colour: text colour (default: "black")
- angle: rotation angle in degrees (default: 0)
- hjust: horizontal justification, 0-1 (default: 0.5)
- vjust: vertical justification, 0-1 (default: 0.5)
- alpha: transparency (default: 1)

See Also

markdown_grob, geom_latex

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  df <- data.frame(
    x = 1:3, y = 1:3,
    lab = c("**bold**", "*slope* $\\beta_1$", "`code` and $x^2$")
  )
  ggplot(df, aes(x, y, label = lab)) + geom_markdown()
}
```

---

grobMark *Position of a named point in a LaTeX grob*

Description

Returns the position of a `\mark{name}` written in the LaTeX, as grid units that can be passed straight to other grid functions, for example to point an arrow at part of a formula.

Usage

```
grobMark(grob, name)
```

Arguments

grob	A grob from latex_grob().
name	The name given to \mark{}.

Value

A list with `x` and `y`, each a `grid::unit()`. Rotation (`rot`) is not taken into account.

See Also

latex_grob

Examples

```
g <- latex_grob(r"($a\mark{eq}^2 = b + c^2$)",
                x = grid::unit(0.5, "npc"),
                y = grid::unit(0.5, "npc"))
grid::grid.newpage(); grid::grid.draw(g)
mk <- grobMark(g, "eq")
grid::grid.points(mk$x, mk$y, pch = 19,
                  gp = grid::gpar(col = "red"))
```

`latex_cache_limit` *Layout cache*

Description

Recently drawn expressions are cached, so drawing the same one again is fast. `latex_cache_limit()` sets how many are kept (512 by default), `latex_cache_clear()` empties the cache, and `latex_cache_info()` reports on it.

Usage

```
latex_cache_limit(n = 512L)

latex_cache_clear()

latex_cache_info()
```

Arguments

`n`	Number of entries to keep. `0` turns the cache off.

Value

`latex_cache_limit()` returns the previous limit and `latex_cache_clear()` returns `NULL`, both invisibly. `latex_cache_info()` returns a list with `size`, `max_size`, `hits` and `misses`.

See Also

`latex_grob`, `latex_options`

Examples

```
latex_cache_limit(256)
grid.latex("$e^{i\\pi} + 1 = 0$")
latex_cache_info()
latex_cache_clear()
```

latex_dims *Size of a LaTeX expression*

Description

Size of a LaTeX expression

Usage

```
latex_dims(
  tex,
  math_font = "",
  max_width = 0,
  tex_style = "",
  input_mode = c("mixed", "math", "document"),
  render_mode = c("typeface", "path"),
  justify = FALSE,
  line_break = c("greedy", "optimal"),
  gp = grid::gpar()
)
```

Arguments

tex	LaTeX, as a character string.
math_font	Math font: "lete" (Lete Sans Math, the default), "stix" (STIX Two Math), or one added with load_font(). See available_fonts(). A font that is not loaded, or has no math table, is an error.
max_width	Width in big points (1/72 inch) at which lines wrap. 0, the default, does not wrap.
tex_style	Force a TeX style: "display", "text", "script" or "scriptscript". "", the default, follows the delimiters. See Details.
input_mode	How tex is read: • "mixed" (default): text, with math between $...$ or \(...\). A newline starts a new line. • "math": everything is math; write text in \text{}. • "document": a LaTeX document body, with paragraphs, numbered headings and displayed equations. Use it with max_width.
render_mode	"typeface" (default) draws glyphs as text, which can be selected in PDF and SVG output. It needs a device such as ragg, svglite or grDevices::cairo_pdf(), and falls back to "path" on others, such as pdf(). "path" draws glyphs as outlines, which works on every device.
justify	If TRUE, wrapped lines are stretched to fill max_width, except the last. Needs max_width.
line_break	"greedy" (default) fills one line at a time. "optimal" chooses the breaks for the whole paragraph. Needs max_width.

gp — Graphical parameters from grid::gpar(): col, fontfamily, fontsize, cex and lineheight. See Details.

Value

A list of grid units in big points:

- width, height: the size of the bounding box.
- depth: how far it extends below the baseline.
- baseline: the height of the baseline above the bottom.

And is_split: TRUE if the text was wrapped over several lines.

Examples

```
latex_dims(r"($\frac{a}{b}$)")
```

---

latex_grob *Create a grid grob from LaTeX*

Description

latex_grob() returns a grob that draws LaTeX: a formula, a label mixing text and math, or a document body. grid.latex() draws it straight away. The grob works with grobWidth(), grobHeight(), grobX() and grobY().

Usage

```
latex_grob(
  tex,
  x = grid::unit(0.5, "npc"),
  y = grid::unit(0.5, "npc"),
  default.units = "npc",
  hjust = 0.5,
  vjust = 0.5,
  rot = 0,
  math_font = "",
  max_width = 0,
  tex_style = "",
  input_mode = c("mixed", "math", "document"),
  render_mode = c("typeface", "path"),
  justify = FALSE,
  line_break = c("greedy", "optimal"),
  debug = FALSE,
  name = NULL,
  gp = grid::gpar()
)

grid.latex(tex, ...)
```

Arguments

tex	LaTeX, as a character string.
x, y	Position.
default.units	Units for x and y when they are numbers.
hjust, vjust	Justification: a number in [0, 1], or a name. hjust takes "left", "center" and "right" (also "centre", "middle", "bbleft", "bbcentre", "bbright"). vjust takes "bottom", "center", "top" and "baseline", which puts the formula's baseline on y.
rot	Rotation in degrees, counter-clockwise.
math_font	Math font: "lete" (Lete Sans Math, the default), "stix" (STIX Two Math), or one added with load_font(). See available_fonts(). A font that is not loaded, or has no math table, is an error.
max_width	Width in big points (1/72 inch) at which lines wrap. 0, the default, does not wrap.
tex_style	Force a TeX style: "display", "text", "script" or "scriptscript". "", the default, follows the delimiters. See Details.
input_mode	How tex is read: • "mixed" (default): text, with math between $...$ or \(...\). A newline starts a new line. • "math": everything is math; write text in \text{}. • "document": a LaTeX document body, with paragraphs, numbered headings and displayed equations. Use it with max_width.
render_mode	"typeface" (default) draws glyphs as text, which can be selected in PDF and SVG output. It needs a device such as ragg, svglite or grDevices::cairo_pdf(), and falls back to "path" on others, such as pdf(). "path" draws glyphs as outlines, which works on every device.
justify	If TRUE, wrapped lines are stretched to fill max_width, except the last. Needs max_width.
line_break	"greedy" (default) fills one line at a time. "optimal" chooses the breaks for the whole paragraph. Needs max_width.
debug	If TRUE, draws the bounding box, the baseline (red) and a dot at the origin of each glyph.
name	Grob name.
gp	Graphical parameters from grid::gpar(): col, fontfamily, fontsize, cex and lineheight. See Details.
...	Arguments passed to latex_grob().

Details

Style:
$...$ sets a formula in text style, as in a paragraph; $$...$$ and \[...\] set it in display style, with larger operators and limits above and below. A label with no delimiters is set in text style. tex_style forces one style on the whole formula; "script" and "scriptscript" are the smaller styles of sub- and superscripts. For part of a formula, use \displaystyle, \textstyle, \scriptstyle or \scriptscriptstyle.

Graphical parameters:

- `col`: the colour. `\textcolor` overrides it.
- `fontfamily`: the font of text, such as `"serif"` or the name of any installed font. Math uses `math_font`. Bold and italic come from `\textbf{}` and `\textit{}`, not from `fontface`.
- `fontsize`, `cex`: the size is `fontsize * cex` points (default 20).
- `lineheight`: line spacing (default 1.2).

Errors:

Invalid LaTeX is drawn as well as it can be, with one warning listing each problem by line and column. An unknown command is drawn in red. A macro that expands without end, or nesting deeper than 400 levels, is an error.

Pasted LaTeX:

LaTeX from a document can be pasted as it is: the output of `knitr::kable()` or `xtable`, a `table` float, or a paper's body with `input_mode = "document"`. The preamble, `\maketitle` and `\label` draw nothing, and `\caption` is drawn where it is written. `\ref` and `\pageref` draw `??`, `\eqref` draws `(??)` and `\cite` draws `[?]`, with a warning. A footnote is set where it is written, and equations are not numbered.

Not supported: `\tag`, `\verb`, `\textsc`, switches such as `\bfseries` and `\itshape` (use `\textbf{}` and `\textit{}`, or `\bf` and `\it`), the `description` list, theorem environments and TikZ.

Images:

`\includegraphics[options]{file}` draws a local PNG, JPEG or SVG file. Each format needs a suggested package: png, jpeg, or rsvg and grImport2 for SVG. `width`, `height` and `scale` size the image, `keepaspectratio` fits it inside both, and `angle` rotates it; `\textwidth` means `max_width`. `trim` and `clip` are ignored with a warning. The extension may be left off, and `\graphicspath{{dir/}}` adds a folder to search.

An SVG stays sharp at any size; a PNG or JPEG warns when it is shown below 150 dpi. PDF and EPS files are not supported. A file that cannot be drawn is an error, except with `input_mode = "document"`, where it warns and draws the file name instead.

Parallel code:

Do not draw in forked processes, such as `parallel::mclapply()` or `future::plan(multicore)`. Use `future::plan(multisession)` or a socket cluster instead.

Value

`latex_grob()` returns a grob of class `"latexgrob"`; `grid.latex()` draws it and returns it invisibly.

See Also

`latex_dims()`, `latex_options()`, `markdown_grob()`, `geom_latex()`

Examples

```
grid::grid.newpage()
grid.latex(r"($x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$)",
           y = 0.8, gp = grid::gpar(fontsize = 24))
```

```
  # Colour, and a rotated grob
  g <- latex_grob(r"($\colorbox{BurntOrange}{x^{2}} + y^{2}$)",
                  x = 0.3, y = 0.4, rot = 45)
  grid::grid.draw(g)
  grid.latex(r"($\textcolor{red}{x^{2}} + y^{2} = z^{2}$)",
             x = 0.7, y = 0.4, gp = grid::gpar(col = "grey30"))

  # A document body, wrapped at 250 points
  grid::grid.newpage()
  doc <- r"(\section{Results}
The fitted line is
\[ \hat{y} = \beta_0 + \beta_1 x, \]
and its slope, $\beta_1$, is positive.)"
  grid.latex(doc, input_mode = "document", max_width = 250,
             x = 0.05, y = 0.95, hjust = 0, vjust = 1,
             gp = grid::gpar(fontsize = 12))
```

`latex_options` *Set or show default rendering options*

Description

Sets defaults for `latex_grob()`, `grid.latex()`, `latex_dims()`, `latex_tree()` and the markdown functions. An argument given in a call always wins over the default.

Usage

```
latex_options(
  math_font = NULL,
  render_mode = NULL,
  tex_style = NULL,
  input_mode = NULL,
  justify = NULL,
  line_break = NULL,
  markdown_style = NULL,
  device_math = NULL,
  main_font = NULL,
  sans_font = NULL,
  mono_font = NULL
)

reset_latex_options()
```

Arguments

`math_font`	Math font: a loaded font with a math table; see `available_fonts()`.
`render_mode`	`"typeface"` or `"path"`; see `latex_grob()`.

`tex_style` "", "display", "text", "script" or "scriptscript"; see `latex_grob()`.

`input_mode` "mixed", "math" or "document"; see `latex_grob()`.

`justify` If TRUE, stretch wrapped lines to fill `max_width`.

`line_break` "greedy" or "optimal"; see `latex_grob()`.

`markdown_style` Default style for `markdown_grob()` and `markdown_box_grob()`: a `markdown_style()`, CSS text, or a path to a `.css` file.

`device_math` If TRUE, any plot label containing math, such as "Slope x^2", is typeset as LaTeX. This works for base graphics (`main`, `xlab`, `text()`, `legend()`, ...), and also for lattice, grid and ggplot2. Labels without math, such as "Cost $5-$10", are left alone.

Heights are not adjusted: a tall formula can overflow a margin or a `legend()` box. Measure it with `latex_dims()` and make room with `par(mar = )`. See `vignette("base-graphics")` for the rules and limitations.

`main_font`, `sans_font`, `mono_font`

The fonts for text: the body, `\textsf` and `\sffamily`, and `\texttt`, `\ttfamily` and `\verb`. Each is the name of a font given to `load_font()`, an installed family, or a font file, which is loaded. NULL keeps the defaults: `gp$fontfamily` (or "sans") for the body, "sans" and "mono". A `gp$fontfamily` in a call wins over `main_font`.

Details

With no arguments, returns the current settings (NULL means the built-in default). Setting an option to NULL resets it. The previous settings are returned, so `do.call(latex_options, old)` restores them.

Size and line spacing are set with gp (`fontsize`, `cex`, `lineheight`), not here.

Value

The previous settings, invisibly. With no arguments, the current settings.

See Also

`available_fonts()`, `load_font()`, `latex_grob()`

Examples

```
old <- latex_options(math_font = "stix", render_mode = "typeface")
grid.latex("$\\sum_{i=1}^{n} i^{2}$", gp = grid::gpar(fontsize = 14))
do.call(latex_options, old)

# Math in base graphics, with no other change to the plotting code.
latex_options(device_math = TRUE)
plot(1:10, (1:10)^2,
     main = "Slope $\\hat{\\beta}_1 = \\sum_{i=1}^{n} x_i^2$",
     ylab = "$y^2$")
```

latex_tree *The layout of a LaTeX expression*

Description

Returns what a formula is drawn from: one row per glyph, line, rectangle or run of text, with its position. Useful for checking alignment or building your own grobs.

Usage

```
latex_tree(
  tex,
  math_font = "",
  max_width = 0,
  tex_style = "",
  input_mode = c("mixed", "math", "document"),
  render_mode = c("typeface", "path"),
  justify = FALSE,
  line_break = c("greedy", "optimal"),
  gp = grid::gpar()
)
```

Arguments

tex	LaTeX, as a character string.
math_font	Math font: "lete" (Lete Sans Math, the default), "stix" (STIX Two Math), or one added with load_font(). See available_fonts(). A font that is not loaded, or has no math table, is an error.
max_width	Width in big points (1/72 inch) at which lines wrap. 0, the default, does not wrap.
tex_style	Force a TeX style: "display", "text", "script" or "scriptscript". "", the default, follows the delimiters. See Details.
input_mode	How tex is read: • "mixed" (default): text, with math between $...$ or \(...\). A newline starts a new line. • "math": everything is math; write text in \text{}. • "document": a LaTeX document body, with paragraphs, numbered headings and displayed equations. Use it with max_width.
render_mode	"typeface" (default) draws glyphs as text, which can be selected in PDF and SVG output. It needs a device such as ragg, svglite or grDevices::cairo_pdf(), and falls back to "path" on others, such as pdf(). "path" draws glyphs as outlines, which works on every device.
justify	If TRUE, wrapped lines are stretched to fill max_width, except the last. Needs max_width.

line_break	"greedy" (default) fills one line at a time. "optimal" chooses the breaks for the whole paragraph. Needs max_width.
gp	Graphical parameters from grid::gpar(): col, fontfamily, fontsize, cex and lineheight. See Details.

Value

A list of class "latex_tree":

- records: a data frame with one row per drawn element, with columns such as type, x, y, glyph, font_size, color and text.
- bbox: width, height, depth and baseline, in big points.
- tex: the input.
- render_mode: the render mode used.

See Also

latex_grob, latex_dims

Examples

```
tree <- latex_tree("\\frac{a}{b}")
print(tree)
head(tree$records)
```

---

latex_wrap *Wrap the text of a label in* \text{}

Description

Turns a label that mixes text and math into pure math: text is wrapped in \text{} and math is kept as it is. latex_grob() does not need this; use it to pass a label to latex_grob(input_mode = "math") or to another renderer that accepts only math.

Math between $...$, $$...$$, \(...\) or \[...\], and math environments, are kept. A newline becomes a line break, and escaped characters such as \$ stay text.

Usage

```
latex_wrap(tex, input_mode = c("mixed", "math"))
```

Arguments

tex	A character vector.
input_mode	"mixed" (default) wraps the text. "math" returns tex unchanged.

Value

A character vector the same length as `tex`.

Examples

```
latex_wrap(r"(The equation \(E=mc^2\) is famous)")
latex_wrap(r"(Cost: \$100 for $x$ items)")
latex_wrap("Line 1\nLine 2")
latex_wrap(r"(\frac{\alpha}{\beta})", input_mode = "math")
```

---

load_font *Load a font*

Description

Registers a font under a name that works everywhere a font is named: `gp = gpar(fontfamily = )`, `latex_options()` (`main_font`, `sans_font`, `mono_font`, `math_font`) and `element_latex(family = )`. The font comes from a file or from an installed family.

Usage

```
load_font(x, name = NULL, bold = NULL, italic = NULL, bolditalic = NULL)
```

Arguments

`x`	A font file (`.otf`, `.ttf` or `.ttc`), or the name of an installed font family.
`name`	The name to register the font under. The default is the font's family name.
`bold`, `italic`, `bolditalic`	Files of the other faces of the same family, so that bold and italic text uses the real design. Only for a file: the faces of an installed family are found automatically. A face left out is drawn with the regular file.

Details

A math font loaded here can also be used as text. A font that is already installed does not need loading to be used as `gp$fontfamily`; load it to give it a short name, or to set it as a role in `latex_options()`.

The text of a loaded font is measured and drawn by gridmicrotex from the font's own file, so it looks the same on every device, base `pdf()` included, with its kerning and ligatures. It is drawn as glyphs where the device has them and as outlines where it has not (and for rotated text). Other families are drawn by the device, as are right-to-left text and a character the font has no glyph for, which the device may find in another font. In base graphics with `latex_options(device_math = TRUE)` the device draws all text, so it has to know the font itself.

Value

The name the font is registered under, invisibly.

See Also

available_fonts(), latex_options()

Examples

```
# The bundled STIX font stands in for your own font file here
otf <- system.file("fonts", "STIXTwoMath-Regular.otf",
                   package = "gridmicrotex")
load_font(otf, name = "My Font")
available_fonts()
```

`markdown_box_grob` *Render a markdown document as a boxed grid grob*

Description

Draws a markdown document (headings, paragraphs, lists, task lists, quotes, code, tables, rules and images) inside an optional box. Text wraps to `width`, and math goes between `$...$`. Everything `markdown_grob()` supports works inside each block.

Usage

```
markdown_box_grob(
  md,
  x = grid::unit(0.5, "npc"),
  y = grid::unit(0.5, "npc"),
  width = grid::unit(1, "npc"),
  height = NULL,
  hjust = 0.5,
  vjust = 0.5,
  halign = 0,
  valign = 1,
  padding = NULL,
  margin = NULL,
  box_gp = NULL,
  r = NULL,
  style = NULL,
  name = NULL,
  gp = grid::gpar(),
  vp = NULL
)
```

Arguments

`md`	Markdown, as a character string.
`x, y`	Position of the box.

width	Width of the box. NULL fits the box to its content, with no wrapping.
height	Height of the box. NULL (default) fits the content.
hjust, vjust	Justification of the box about x and y.
halign	Alignment of blocks in the box: 0 left (default), 0.5 centre, 1 right.
valign	Vertical alignment of the content when height leaves room: 1 top (default), 0 bottom.
padding, margin	A grid::unit() of length 1, or 4 for top, right, bottom and left. Padding is inside the box, margin outside. NULL (default) uses the style's body rule.
box_gp	Fill and border of the box, such as gpar(fill = "grey95", col = "black"). NULL (default) uses the style's body rule, and draws no box if it has none.
r	Corner radius. NULL (default) uses the style's body rule.
style	A markdown_style(), CSS text, or a path to a .css file. NULL (default) uses latex_options("markdown_style"), if set.
name	Grob name.
gp	Graphical parameters for the text. fontsize also scales the spacing between blocks.
vp	A viewport. If given, x, y, width, height, hjust and vjust are ignored.

Details

A table that is too wide for the box wraps its widest columns; table-layout: fixed gives the columns equal widths instead. Code lines do not wrap.

An image on its own line is a block, scaled down to fit; an image inside a sentence is inline. Images must be local PNG, JPEG or SVG files, and need the png or jpeg package, or rsvg and grImport2 for SVG. An image that cannot be drawn is an error.

Value

A grob of class "markdownbox".

Styling

style sets the look of each tag:

```
markdown_box_grob(md, style = markdown_style(
  h1         = md_style(color = "steelblue", font_size = 2),
  blockquote = md_style(border_left = "3px solid grey60")
))
```

Or the same as CSS, as text or a .css file:

```
markdown_box_grob(md, style = "
  h1 { color: steelblue; font-size: 2rem }
  blockquote { border-left: 3px solid grey60 }
")
```

To style one part, wrap it in a `<div>` with a `class` or `style`, with blank lines around the tags:

```
<div class="note">

## This heading only

</div>
```

`<span class="...">` works the same inside a line. See `markdown_style()` for tag names and `md_style()` for properties.

See Also

`markdown_grob`, `latex_grob`

Examples

```
md <- paste(
  "# Results", "",
  "The slope is $\\beta_1$ with *p* < 0.001.", "",
  "- first point", "- second point",
  sep = "\n"
)
grid::grid.newpage()
grid::grid.draw(markdown_box_grob(
  md,
  width = grid::unit(4, "in"),
  padding = grid::unit(8, "pt"),
  box_gp = grid::gpar(fill = "grey95", col = "grey40")
))
```

---

markdown_grob *Render a markdown label as a grid grob*

Description

Draws a line of markdown, such as a plot title, with LaTeX math between `$...$`. `**bold**`, `*italic*`, `` `code` `` and `~~strike~~` work, and so does some inline HTML. For headings, lists, tables and other blocks, use `markdown_box_grob()`.

Usage

```
markdown_grob(md, style = NULL, ...)

grid.markdown(md, ...)
```

Arguments

md	Markdown, as a character string.
style	A markdown_style(), CSS text, or a path to a .css file. NULL (default) uses latex_options("markdown_style"), if set. Only properties marked *inline* in md_style() apply.
...	Passed to latex_grob(), such as x, y, hjust, vjust, rot, max_width and gp.

Details

These HTML tags are supported:

tag	effect
<b>,	bold
<i>, , <cite>, <dfn>, <var>, <address>	italic
<code>, <kbd>, <samp>, <tt>	monospace
<u>, <ins>	underline
<s>, <del>, <strike>	strikethrough
<sub>, <sup>	sub / superscript
<mark>	yellow highlight
<small>, <big>	smaller / larger
<q>	quotation marks
 	line break
	see below

A style attribute can set color, text-decoration (underline, line-through), font-size (such as 12pt, 1.2em or smaller) and font-family; other properties are ignored. Colours are R colour names, CSS names, #rgb, #rrggbb or rgb(). font-family takes serif, sans-serif, monospace, an installed font, or a font named with load_font().

Tags nest and can hold markdown and math. Other tags are dropped and their text kept. Links keep their text only. Images must be local PNG, JPEG or SVG files. The characters U+E000 to U+E002 are removed from the input.

Value

A grob, as from latex_grob(). grid.markdown() draws it and returns it invisibly.

See Also

markdown_box_grob(), geom_markdown()

Examples

```
grid::grid.newpage()
grid.markdown(r"(The **fitted** slope is $\beta_1$, *p* < 0.001)")
```

`markdown_style` *A style for markdown rendering*

Description

Creates a style for `markdown_grob()`, `markdown_box_grob()` and the ggplot2 markdown functions, from CSS, a preset, or `md_style()` declarations.

Usage

```
markdown_style(base = NULL, css = NULL, ...)
```

Arguments

`base`	`NULL` (default) for the built-in style, a preset name such as `"github"`, or a `markdown_style` to extend.
`css`	CSS text, or a path to a `.css` file.
`...`	Tag styles, each an `md_style()`. Start a name with a dot for a class, as in `.note = md_style(...)`.

Details

Tags are named as in HTML: `body`, `p`, `h1` to `h6`, `ul`, `ol`, `li`, `blockquote`, `pre` (code block), `code` (inline code), `strong`, `em`, `table`, `tr`, `td`, `th`, `hr`, `img`, `a` (link), `math` (a `$$...$$` paragraph), `footnote`, `div` and `span`. Every tag inherits from `body`, and table cells inherit from `table`.

As in CSS, an inline `style` beats a class (`.note`), which beats a tag (`h1`), and a later rule beats an earlier one. Colour, font, line height and alignment are inherited by nested blocks; margins, padding and borders are not.

Tag selectors, class selectors and lists (`h1, h2`) are supported. Other selectors are ignored.

Value

A style object of class `"gridmicrotex_markdown_style"`.

See Also

`md_style`, `markdown_box_grob`

Examples

```
markdown_style()
markdown_style("github", h1 = md_style(color = "firebrick"))
markdown_style(css = "h1 { color: steelblue } .note { padding-left: 2em }")
```

`md_style` *Declarations for one markdown tag*

Description

CSS properties for one tag, for use in `markdown_style()`. Names are CSS property names with `_` for `-`, so `font_size` sets `font-size`.

Usage

```
md_style(...)
```

Arguments

`...`	Properties, as named arguments.

Details

A length can be a number, meaning a multiple of the body font size (`font_size = 2.5`); a CSS string such as `"2.5em"` or `"12pt"`; or a `grid::unit()`.

The supported properties are below. *Inline* ones also work on a `<span>` and in `markdown_grob()`; *block* ones need `markdown_box_grob()`.

property	scope	notes
`color`	inline + block	
`font_size`	inline + block	
`font_family`	inline + block	
`font_weight`	inline + block	
`font_style`	inline + block	
`text_decoration`	inline + block	`underline`, `overline`, `line-through`
`background`	inline + block	a fill behind the text
`border`	inline	a frame; the inset is fixed
`border_style`	inline	only `double`
`border_radius`	inline	rounds the frame
`box_shadow`	inline	
`visibility`	inline	`hidden` keeps the space
`vertical_align`	inline	`super`, `sub`, or a length
`transform`	inline	`rotate()`, `scale()`, `scaleX(-1)`
`line_height`	block	unitless, as `gpar()` wants it
`margin_top`, `margin_bottom`	block	margins do not collapse
`margin_left`, `margin_right`	block	
`padding_left`, `padding_right`	block	
`padding_top`, `padding_bottom`	block	
`margin`, `padding`	block	the CSS shorthand: one to four lengths, in CSS's order
`text_align`	block	`left`, `center`, `right`
`border_left`	block	the `blockquote` bar
`border_top`	block	the `hr` rule

`border_bottom`	`tr`	a rule under each table row
`border_color`	`table`	colour of the table's rules
`table_layout`	`table`	`fixed` divides the width evenly between the columns
`height`	block	the band an `hr` sits in
`bullet`	`ul`	raw LaTeX for the marker glyph
`marker_gap`	`ul`, `ol`	marker to text

`font_size` also takes the keywords `xx-small` to `xx-large`, `smaller` and `larger`.

On `body`, `background`, `border`, `border_radius`, `padding` and `margin` style the box of `markdown_box_grob()` or of an `element_markdown()` title:

```
body { background: grey95; padding: 8px;
       border: 1px solid grey60; border-radius: 4px }
```

The `box_gp`, `padding`, `margin` and `r` arguments of `markdown_box_grob()` override this rule.

An unknown property is an error here, but is ignored in CSS text. Small caps, `font-variant-numeric` and padding inside an inline `border` are not supported.

Value

An object of class `gridmicrotex_md_style`.

See Also

`markdown_style`, `markdown_box_grob`

Examples

```
md_style(color = "steelblue", font_size = 2.5, margin_top = 1.2)
```

---

register_highlighter *Add a syntax highlighting grammar*

Description

Adds a language for highlighting fenced code blocks in `markdown_box_grob()`, from a KDE syntax file (the format used by Kate and Pandoc). The easiest start is a copy of a built-in grammar:

Usage

```
register_highlighter(lang, file)
```

Arguments

`lang`	Language name, as written after the opening fence. Case is ignored. A built-in language of the same name is replaced.
`file`	Path to a KDE syntax XML file.

Details

```
file.copy(system.file("highlight", "python.xml",
                      package = "gridmicrotex"),
          "mylang.xml")
```

Many files from https://kate-editor.org/syntax/ work as they are. Files that use *dynamic* rules are refused with an error. Those files are mostly GPL or LGPL licensed; check the licence before you redistribute one.

Colours come from markdown_style(), using Pandoc's class names (kw keyword, co comment, st string, ...), so a grammar needs no colours of its own.

Value

lang, invisibly.

See Also

available_highlighters, markdown_box_grob, markdown_style

Examples

```
# Registering a grammar under a name of your own.
f <- system.file("highlight", "python.xml", package = "gridmicrotex")
register_highlighter("mypython", f)
"mypython" %in% available_highlighters()
```

Index

∗ **datasets**
    geom_latex, 8
    geom_markdown, 10

aes(), *8*, *11*
annotation_borders(), *9*, *12*
available_fonts, 2
available_fonts(), *5*, *15*, *17*, *19–21*, *24*
available_highlighters, 3, *31*

clear_macros *(*define_macro*)*, 4

define_macro, 4

element_latex, 5, *7*
element_markdown, 6
element_markdown(), *30*

fortify(), *8*, *11*

geom_latex, 8, *12*
geom_latex(), *18*
geom_markdown, 10
geom_markdown(), *27*
GeomLatex *(*geom_latex*)*, 8
GeomMarkdown *(*geom_markdown*)*, 10
ggplot(), *8*, *11*
ggplot2::element_text(), *5–7*
ggplot2::geom_text(), *8*, *10*
ggplot2::layer(), *9*, *12*
grDevices::cairo_pdf(), *6*, *9*, *15*, *17*, *21*
grid.latex *(*latex_grob*)*, 16
grid.latex(), *19*
grid.markdown *(*markdown_grob*)*, 26
grid::gpar(), *16*, *17*, *22*
grid::unit(), *7*, *13*, *25*, *29*
grobMark, 13

latex_cache_clear *(*latex_cache_limit*)*, 14
latex_cache_info *(*latex_cache_limit*)*, 14
latex_cache_limit, 14
latex_dims, 15, *22*
latex_dims(), *18–20*
latex_grob, *4*, *13*, *14*, 16, *22*, *26*
latex_grob(), *12*, *13*, *19*, *20*, *22*, *27*
latex_options, *4*, *14*, 19
latex_options(), *2*, *3*, *7*, *18*, *23*, *24*
latex_tree, 21
latex_tree(), *19*
latex_wrap, 22
layer position, *9*, *11*
layer stat, *9*, *11*
list_macros *(*define_macro*)*, 4
load_font, 23
load_font(), *2*, *3*, *5*, *15*, *17*, *20*, *21*, *27*

markdown_box_grob, 24, *28*, *30*, *31*
markdown_box_grob(), *3*, *7*, *20*, *26–30*
markdown_grob, *7*, *12*, 26, *26*
markdown_grob(), *10*, *18*, *20*, *24*, *28*, *29*
markdown_style, 28, *30*, *31*
markdown_style(), *7*, *12*, *20*, *25–27*, *29*, *31*
md_style, *28*, 29
md_style(), *12*, *26–28*

register_highlighter, *4*, 30
register_highlighter(), *3*
reset_latex_options *(*latex_options*)*, 19