Package: gridmicrotex (via r-universe)

May 21, 2026

Title Native 'LaTeX' Math Rendering for Grid Graphics

Version 0.0.4

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.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

Depends R (>= 4.2.0)

LinkingTo Rcpp, systemfonts

Imports grDevices, grid, Rcpp, systemfonts, tools

Suggests ggplot2 (>= 4.0.0), knitr, ragg, rmarkdown, S7, testthat (>= 3.0.0), 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

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev pkg-config

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

Date/Publication 2026-05-21 22:05:37 UTC

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

RemoteRef HEAD

RemoteSha 154f04ffa026c44fece2e056818e8c1ab17c8f17

Contents

`available_math_fonts` *List available math fonts*

Description

Returns the names of all math fonts currently loaded by MicroTeX. These names can be passed to the `math_font` parameter of `latex_grob` and `grid.latex`.

Usage

```
available_math_fonts()
```

Value

A character vector of math font names.

Font pairing

The bundled math fonts have different styles. For a consistent look, pair them 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"`

Additional math fonts can be loaded with `load_font`.

Examples

```
available_math_fonts()
```

check_fonts *Check font status*

Description

Reports which math fonts are loaded and available for rendering. Shows the MicroTeX version, loaded math fonts, and whether bundled font files are present.

Usage

```
check_fonts()
```

Value

Invisibly returns the character vector of available font names.

Examples

```
check_fonts()
```

define_macro *Define a user-level LaTeX macro*

Description

Registers a zero-argument shorthand that is expanded by text substitution before the expression reaches the MicroTeX parser. Useful for domain-specific notation (e.g. `\RR` for `\mathbb{R}`) you reuse across many plots.

Usage

```
define_macro(name, definition)

clear_macros(name = NULL)

list_macros()
```

Arguments

`name`	Macro name **without** the leading backslash. For `clear_macros`, the macro name to drop, or `NULL` (default) to clear all.
`definition`	LaTeX source the macro expands to.

Value

- `define_macro`: Invisibly returns `NULL`.
- `clear_macros`: Invisibly returns `NULL`.
- `list_macros`: A named character vector mapping macro names to their expansions. Empty if no macros are defined.

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

Use this as a theme element for axis titles, axis labels, plot titles, or any other text element in a ggplot2 theme. The text string is parsed as LaTeX math and rendered via MicroTeX.

Usage

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

Arguments

`math_font`	Name of the math font to use (e.g., `"stix"`). Use `""` (default) for Lete Sans Math, which pairs with R's default sans-serif text font. See `available_math_fonts` for loaded fonts.
`fontsize`	Convenience alias for `size`; when supplied, it is forwarded to `ggplot2::element_text()` as the text size in points. If `NULL` (default), the theme's inherited size is used.

`lineheight`	Multi-line height multiplier (default 1.2), matching `grid::gpar()` semantics.
`max_width`	Numeric maximum width in big points for automatic line wrapping. Use `0` (default) for no wrapping.
`input_mode`	How `tex` is interpreted before being parsed. `"mixed"` wraps the input in `\text{...}` so the string reads as ordinary text and `$...$` (or `\(...\)`) opens math mode, matching document-level LaTeX semantics. Useful for labels that arrive from external sources mixing prose and math without explicit `\text{}` markers. `"math"` (default) is the standard MicroTeX behaviour — the whole string is treated as math, so unwrapped prose renders as spaced math italics. The default can be changed globally via `latex_options(input_mode = "mixed")`. See `latex_wrap` for details on the wrapping process.
`render_mode`	Character string: `"typeface"` (default) renders glyphs as native text using the math font, producing selectable/accessible text in PDF and SVG output. Bundled math fonts and any registered via `load_font` are read directly from their OTF files — no system-wide font install is required. Falls back to path mode automatically on devices that lack the R $\geq$ 4.3 glyph engine (e.g., the base `pdf()` device). For selectable PDF output, prefer `cairo_pdf`. `"path"` renders math symbols as filled vector paths (works on all devices but text is not selectable in PDF/SVG).
`...`	Additional arguments passed to `ggplot2::element_text()` (e.g., `size`, `colour`, `hjust`).

Details

Dollar signs (`$...$`) in the label text are stripped automatically so that both `"\frac{a}{b}"` and `"$\frac{a}{b}$"` work.

This element is an S7 subclass of `ggplot2::element_text`, so it inherits all standard text properties (size, colour, hjust, etc.) from the theme and supports `merge_element()` correctly.

Value

An S7 object of class `element_latex`, inheriting from `ggplot2::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())
}
```

geom_latex *A ggplot2 geom for LaTeX math labels*

Description

Renders LaTeX math expressions as native grid grobs within a ggplot2 plot. Each label is parsed and laid out by MicroTeX, producing resolution-independent vector output.

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"),
  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.

`...` Other arguments passed to `layer`.

`fontsize` Default font size in points. Overridden by the `size` aesthetic if mapped.

`math_font` Name of the math font to use (e.g., `"stix"`).

`lineheight` Multi-line height multiplier (default 1.2), matching `grid::gpar()` semantics.

`max_width` Maximum width in big points for automatic line wrapping (default: 0, no wrapping).

`input_mode` How `tex` is interpreted before being parsed. `"mixed"` wraps the input in `\text{...}` so the string reads as ordinary text and `$...$` (or `\(...\)`) opens math mode, matching document-level LaTeX semantics. Useful for labels that arrive from external sources mixing prose and math without explicit `\text{}` markers. `"math"` (default) is the standard MicroTeX behaviour — the whole string is treated as math, so unwrapped prose renders as spaced math italics. The default can be changed globally via `latex_options(input_mode = "mixed")`. See `latex_wrap` for details on the wrapping process.

`render_mode` Character string: `"typeface"` (default) renders glyphs as native text using the math font, producing selectable/accessible text in PDF and SVG output. Bundled math fonts and any registered via `load_font` are read directly from their OTF files — no system-wide font install is required. Falls back to path mode automatically on devices that lack the R $\geq$ 4.3 glyph engine (e.g., the base `pdf()` device). For selectable PDF output, prefer `cairo_pdf`. `"path"` renders math symbols as filled vector paths (works on all devices but text is not selectable in PDF/SVG).

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`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 math 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"($\hat{y} = \beta_0 + \beta_1 x$)")
}
```

---

grobMark *Look up a named anchor inside a LaTeX grob*

Description

Resolves a `\mark{name}` that was placed inside the LaTeX source to a pair of **grid** units in the grob's parent viewport. The returned units already account for the grob's viewport position and `hjust`/`vjust`, so you can pass them directly to grid drawing functions to anchor other graphics on parts of the formula.

Usage

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

Arguments

grob	A latexgrob returned by latex_grob.
name	The mark name (the argument to \mark{...}).

Value

A list with elements x and y, each a unit. Mark coordinates are evaluated in the grob's parent viewport.

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 *Set the maximum number of entries kept in the LaTeX layout cache*

Description

The cache stores parsed layout information for recently rendered LaTeX expressions, keyed by the expression and relevant rendering parameters (font, size, macros, etc.). This speeds up repeated rendering of the same expressions, especially in loops or interactive sessions. The default limit is 512 entries, which should be sufficient for most use cases. When the limit is exceeded, the least recently used entries are automatically evicted.

Usage

```
latex_cache_limit(n = 512L)

latex_cache_clear()

latex_cache_info()
```

Arguments

n	Non-negative integer cache capacity. Default is 512. Set to 0 to disable caching.

Value

- latex_cache_limit: Invisibly returns the previous limit.
- latex_cache_clear: Invisibly returns NULL.
- latex_cache_info: A list with elements size (entries currently stored), 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 *Get dimensions of a LaTeX expression*

Description

Get dimensions of a LaTeX expression

Usage

```
latex_dims(
  tex,
  math_font = "",
  max_width = 0,
  tex_style = "",
  input_mode = c("mixed", "math"),
  render_mode = c("typeface", "path"),
  gp = grid::gpar()
)
```

Arguments

tex	Character string of LaTeX math code.
math_font	Name of the math font to use (e.g., "stix"). Use "" (default) for Lete Sans Math, which pairs with R's default sans-serif text font. See available_math_fonts for loaded fonts.
max_width	Numeric maximum width in big points for automatic line wrapping. Use 0 (default) for no wrapping.
tex_style	Character: TeX style override. One of "" (default; let the parser decide), "display", "text", "script", or "scriptscript". See latex_grob for the semantics of each value.
input_mode	How tex is interpreted before being parsed. "mixed" wraps the input in \text{...} so the string reads as ordinary text and $...$ (or \(...\)) opens math mode, matching document-level LaTeX semantics. Useful for labels that arrive from external sources mixing prose and math without explicit \text{} markers. "math" (default) is the standard MicroTeX behaviour — the whole string is treated as math, so unwrapped prose renders as spaced math italics. The default can be changed globally via latex_options(input_mode = "mixed"). See latex_wrap for details on the wrapping process.
render_mode	Character string: "typeface" (default) renders glyphs as native text using the math font, producing selectable/accessible text in PDF and SVG output. Bundled math fonts and any registered via load_font are read directly from their OTF files — no system-wide font install is required. Falls back to path mode automatically on devices that lack the R $\geq$ 4.3 glyph engine (e.g., the base pdf() device). For selectable PDF output, prefer cairo_pdf. "path" renders math symbols as filled vector paths (works on all devices but text is not selectable in PDF/SVG).
gp	Graphical parameters (see gpar). Common entries: col (formula foreground), fontfamily / fontface (text font), fontsize / cex (formula size), and lineheight (multi-line spacing). See latex_grob for how each of these flows through MicroTeX.

Value

A list with the following elements:

- width, height, depth: grid unit objects in big points. height is total height (ascent + descent).
- baseline: grid unit object giving the baseline position measured in big points from the *bottom* of the bounding box. Equivalent to height - depth for single-line formulas. Useful for aligning a formula's baseline with surrounding text.
- is_split: logical; TRUE if the formula was wrapped across multiple lines (only possible when max_width > 0).

Examples

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

latex_grob *Create a grid grob from a LaTeX expression*

Description

Parses a LaTeX math expression and returns a grid grob object that renders the formula using native grid graphics primitives. The grob supports standard grid queries such as `grobWidth()`, `grobHeight()`, `grobX()`, and `grobY()`.

A convenience wrapper that creates a `latex_grob` and immediately draws it on the current device via `grid.draw`.

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"),
  render_mode = c("typeface", "path"),
  debug = FALSE,
  name = NULL,
  gp = grid::gpar()
)

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

Arguments

`tex`	Character string of LaTeX math code.
`x, y`	Position in grid coordinates.
`default.units`	Units for x, y if given as numeric.
`hjust, vjust`	Horizontal/vertical justification. Accepts the usual numeric values in `[0, 1]`. As a convenience, `hjust` also accepts the strings `"left"`/`"bbleft"`, `"center"`/`"centre"`/`"middle"`/`"bbcentre"`, and `"right"`/`"bbright"`; `vjust` accepts `"bottom"`, `"center"`/`"centre"`/`"middl` `"top"`, and `"baseline"`. `"baseline"` aligns the formula's math baseline with the anchor point — handy for placing a formula in flowing text.
`rot`	Rotation angle in degrees, counter-clockwise (default: 0). Matches the `rot` parameter of `textGrob`.

math_font	Name of the math font to use (e.g., "stix"). Use "" (default) for Lete Sans Math, which pairs with R's default sans-serif text font. See available_math_fonts for loaded fonts.
max_width	Numeric maximum width in big points for automatic line wrapping. Use 0 (default) for no wrapping.
tex_style	Character: TeX style override. One of "" (default; let the parser decide), "display", "text", "script", or "scriptscript". See latex_grob for the semantics of each value.
input_mode	How tex is interpreted before being parsed. "mixed" wraps the input in \text{...} so the string reads as ordinary text and $...$ (or \(...\)) opens math mode, matching document-level LaTeX semantics. Useful for labels that arrive from external sources mixing prose and math without explicit \text{} markers. "math" (default) is the standard MicroTeX behaviour — the whole string is treated as math, so unwrapped prose renders as spaced math italics. The default can be changed globally via latex_options(input_mode = "mixed"). See latex_wrap for details on the wrapping process.
render_mode	Character string: "typeface" (default) renders glyphs as native text using the math font, producing selectable/accessible text in PDF and SVG output. Bundled math fonts and any registered via load_font are read directly from their OTF files — no system-wide font install is required. Falls back to path mode automatically on devices that lack the R $\geq$ 4.3 glyph engine (e.g., the base pdf() device). For selectable PDF output, prefer cairo_pdf. "path" renders math symbols as filled vector paths (works on all devices but text is not selectable in PDF/SVG).
debug	Logical; if TRUE, draws diagnostic overlays on the grob — the full bounding box (dashed gray), the baseline (solid red), the depth line (dashed gray), and a small dot at each MicroTeX draw record's origin. Useful for checking positioning and diagnosing vertical alignment.
name	Optional grob name.
gp	Graphical parameters (see gpar). Common entries: col (formula foreground), fontfamily / fontface (text font), fontsize / cex (formula size), and lineheight (multi-line spacing). See latex_grob for how each of these flows through MicroTeX.
...	Additional arguments passed to latex_grob.

Details

Controlling TeX style with tex_style:

tex_style selects the size-and-spacing regime MicroTeX applies to the whole expression. It changes the *style* (display vs. text), not the font size — size is always set via gp$fontsize / gp$cex; style-dependent shrinking (for "script" and "scriptscript") is applied on top of that size.

- "" (default): let the parser choose based on the delimiters in tex. Inline delimiters (single $, or \(...\)) produce "text" style; display delimiters (double $$, or \[...\]) produce "display" style. If the string has no delimiters, MicroTeX defaults to "text" style.

- `"display"`: force display style. Large operators (`\sum`, `\int`, `\prod`) render at their full size, limits are placed above/below rather than as subscripts/superscripts, and fractions use full-size numerators and denominators. Useful when you want a display-style equation inline in a label, legend, or `element_latex()` title.
- `"text"`: force text (inline) style. Big operators shrink to their inline size and limits attach as scripts. The right choice for formulas embedded in a line of prose.
- `"script"`: force script style — the size normally used for first-level subscripts and superscripts. Produces a smaller, tighter layout; mainly useful for callouts or sub-labels where a compact equation is wanted.
- `"scriptscript"`: force scriptscript style — the smallest style, used by TeX for doubly-nested scripts. Rarely needed on its own; primarily for very dense annotations.

`tex_style` applies to the entire expression. To override the style of a sub-expression from within `tex`, use the inline TeX commands `\displaystyle`, `\textstyle`, `\scriptstyle`, or `\scriptscriptstyle`.

Graphical parameters (`gp`)**:**

- `col`: default foreground color for the formula. Individual elements can still be overridden with an inline `\textcolor` command in the LaTeX string.
- `fontfamily` / `fontface`: control the appearance of text inside `\text` and `\mbox` blocks. For example, `gpar(fontfamily = "serif")` renders `\text` content in R's serif family. Any font available to R's graphics system works — base families (`"sans"`, `"serif"`, `"mono"`) as well as fonts registered via **showtext** or **systemfonts**. Math symbols always use the selected math font (see `math_font`).
  `fontfamily` *also* drives MicroTeX's layout metrics for non-math text: the matching system font is resolved via **systemfonts**, a minimal metrics file is generated on first use and cached under `tools::R_user_dir("gridmicrotex", "cache")`, so MicroTeX's spacing of `\text` blocks stays in sync with what **grid** actually draws. When `fontfamily` is unset, the R default (`"sans"`) is used. No manual font loading is required for text fonts; `load_font()` remains only for adding custom **math** fonts.
- `fontsize` / `cex`: formula size is `fontsize * cex` big points (default 20 * 1). Both math and text scale together. The effective size is baked into the parsed layout, so downstream viewports that inherit `cex` will not re-scale the grob (matching `textGrob` semantics when `gp` is set explicitly).
- `lineheight`: controls multi-line spacing (default 1.2). The inter-line gap is `(lineheight - 1) * fontsize` big points.

LaTeX document-level wrappers:

The parser accepts raw output from `print.xtable()`, `knitr::kable()`, and similar functions that emit complete `tabular` LaTeX. The following document-level constructs are recognized and rewritten silently before the input reaches MicroTeX:

Removed (no visual effect):

- `%`-to-end-of-line comments (escaped `\%` is preserved)
- preamble: `\documentclass[...]{...}`, `\usepackage[...]{...}`, `\begin{document}` / `\end{document}`
- title metadata: `\maketitle`, `\title{...}`, `\author{...}`
- cross-reference labels: `\label{...}`

- float wrappers: `\begin{table}` / `\end{table}`, `\begin{figure}` / `\end{figure}` (and starred variants)
- layout scopes: `\centering`, `\raggedright`, `\raggedleft`, `\flushleft`, `\flushright`

Rewritten:

- booktabs rules: `\toprule`, `\midrule`, `\bottomrule`, `\cmidrule` are mapped to `\hline`. The optional column-range and trim arguments of `\cmidrule` are discarded (MicroTeX has no concept of partial-column rules).
- `\caption[short]{X}` is extracted as `\text{X}\\` at its source position. The caption renders where it appears in the input (typically below the `tabular` for `xtable`, above for `kable`); expect a slight visual difference from full LaTeX, which positions the caption above or below the float regardless of source order.

Anything not in this list is passed to MicroTeX unchanged. Unknown commands render as literal text, which is useful for spotting unsupported markup.

Parallelism:

The MicroTeX engine keeps mutable C++ state for font caching and text measurement. Rendering is safe single-threaded and under separate-process backends such as `future::plan(multisession)`. It is *not* safe under forked backends (`parallel::mclapply()`, `future::plan(multicore)`) on Unix, because forked workers share that state without synchronisation. Use a socket/multisession backend instead.

Value

A `grid` grob of class `"latexgrob"`.

Invisibly returns the grob.

See Also

`grid.latex`, `latex_dims`, `geom_latex`, `available_math_fonts`, `latex_wrap`, `latex_options`

Examples

```
g <- latex_grob(r"($\fcolorbox{red}{yellow}{\frac{a}{b}}$)",
                x = grid::unit(0.3, "npc"),
                y = grid::unit(0.3, "npc"),
                gp = grid::gpar(fontsize = 30))
grid::grid.draw(g)
# Red formula
grid::grid.draw(latex_grob("$x^{2}$",
                           x = grid::unit(0.3, "npc"),
                           y = grid::unit(0.8, "npc"),
                           gp = grid::gpar(col = "red")))

                           # Rotated formula
grid::grid.draw(latex_grob(r"($\colorbox{BurntOrange}{x^{2}} + y^{2}$)",
                           x = grid::unit(0.6, "npc"),
                           y = grid::unit(0.3, "npc"),
                           gp = grid::gpar(fontsize = 24),
                           rot = 45))
```

```
grid.latex(r"($\textcolor{red}{x^{2}} + y^{2} = z^{2}$)",
           x = grid::unit(0.6, "npc"),
           y = grid::unit(0.8, "npc"),)
```

`latex_options` *Set or query package-wide LaTeX rendering defaults*

Description

A single entry point for project-wide defaults used by `latex_grob`, `grid.latex`, `latex_dims`, and `latex_tree`. Options set here are applied only when the corresponding argument is *not* supplied at the call site, so explicit arguments always win.

Usage

```
latex_options(
  math_font = NULL,
  render_mode = NULL,
  tex_style = NULL,
  input_mode = NULL
)

reset_latex_options()
```

Arguments

`math_font`	Math font name or alias (see `available_math_fonts`).
`render_mode`	Either `"typeface"` or `"path"`.
`tex_style`	TeX style override. One of `""` (let the parser decide), `"display"`, `"text"`, `"script"`, or `"scriptscript"`. `"display"` forces large operators with limits placed over/under, useful for inline labels that should still look like display equations.
`input_mode`	How the input string is interpreted before being handed to MicroTeX. `"math"` (default) treats the whole string as math — the standard MicroTeX behaviour, where letters render as math italics and unwrapped prose looks wrong. `"mixed"` wraps the string in `\text{...}` so it reads as ordinary text, with `$...$` (and `\(...\)`) opening math mode — the document-level LaTeX convention. Useful when consuming labels from other packages that mix prose and math without explicit `\text{}` markers.

Details

Calling `latex_options()` with no arguments returns the current settings (a list whose `NULL` entries mean "use the built-in default"). Supply one or more named arguments to update them.

Font size and line spacing are controlled via gp parameters (`fontsize`, `cex`, `lineheight`) at the grob level — see `latex_grob`.

Value

Invisibly returns the previous settings (a list). With no arguments, returns the current settings visibly.

See Also

available_math_fonts, latex_grob

Examples

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

---

latex_tree *Inspect the parsed layout of a LaTeX expression*

Description

Returns the raw draw-record table produced by MicroTeX's layout pass together with the bounding-box metadata. Useful for debugging alignment issues, building custom grobs on top of the layout, or counting glyphs/paths/rules in a formula.

Usage

```
latex_tree(
  tex,
  math_font = "",
  max_width = 0,
  tex_style = "",
  input_mode = c("mixed", "math"),
  render_mode = c("typeface", "path"),
  gp = grid::gpar()
)
```

Arguments

tex	Character string of LaTeX math code.
math_font	Name of the math font to use (e.g., "stix"). Use "" (default) for Lete Sans Math, which pairs with R's default sans-serif text font. See available_math_fonts for loaded fonts.
max_width	Numeric maximum width in big points for automatic line wrapping. Use 0 (default) for no wrapping.
tex_style	Character: TeX style override. One of "" (default; let the parser decide), "display", "text", "script", or "scriptscript". See latex_grob for the semantics of each value.

`input_mode` How `tex` is interpreted before being parsed. `"mixed"` wraps the input in `\text{...}` so the string reads as ordinary text and `$...$` (or `\(...\)`) opens math mode, matching document-level LaTeX semantics. Useful for labels that arrive from external sources mixing prose and math without explicit `\text{}` markers. `"math"` (default) is the standard MicroTeX behaviour — the whole string is treated as math, so unwrapped prose renders as spaced math italics. The default can be changed globally via `latex_options(input_mode = "mixed")`. See `latex_wrap` for details on the wrapping process.

`render_mode` Character string: `"typeface"` (default) renders glyphs as native text using the math font, producing selectable/accessible text in PDF and SVG output. Bundled math fonts and any registered via `load_font` are read directly from their OTF files — no system-wide font install is required. Falls back to path mode automatically on devices that lack the R $\geq$ 4.3 glyph engine (e.g., the base `pdf()` device). For selectable PDF output, prefer `cairo_pdf`. `"path"` renders math symbols as filled vector paths (works on all devices but text is not selectable in PDF/SVG).

`gp` Graphical parameters (see `gpar`). Common entries: `col` (formula foreground), `fontfamily` / `fontface` (text font), `fontsize` / `cex` (formula size), and `lineheight` (multi-line spacing). See `latex_grob` for how each of these flows through MicroTeX.

Value

A list with class `"latex_tree"` containing:

`records` Data frame of draw records (one row per glyph, path, line, rect, or text block). Columns include `type`, `x`, `y`, `glyph`, `font_size`, `color`, `text`, `codepoint`, `font_file`.

`bbox` Named numeric vector with `width`, `height`, `depth`, `baseline` (all in big points).

`tex` The (macro-expanded) input string.

`render_mode` Rendering mode used for the layout.

See Also

`latex_grob`, `latex_dims`

Examples

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

`latex_wrap` *Wrap standard text for math-first LaTeX renderers*

Description

Parses character strings to safely isolate standard natural language from LaTeX math environments. Standard text is wrapped in `\text{}` blocks, while equations, display math, and specific LaTeX environments are preserved verbatim. This is heavily optimized for passing mixed-content strings (like plot titles or axis labels) to pure-math typesetting engines like MicroTex. The conversion is not perfect, but it should handle most common cases without user intervention.

Usage

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

Arguments

`tex`	`character`. The string or vector of strings to be processed.
`input_mode`	`character`. A length-one character vector dictating the parsing strategy. If `"mixed"` (default), the string is tokenized and text is wrapped. If `"math"`, the parser is bypassed and the string is returned unmodified, assuming the user has provided a pure math equation.

Details

`latex_wrap()` operates as a state-machine tokenizer to ensure that valid LaTeX math is not corrupted by the text-wrapping process. It features:

- **Delimiter Preservation**: Standard inline (`$`, `\(`) and block (`$$`, `\[`) math delimiters are recognized and preserved.
- **Environment Tracking**: Complex nested environments (e.g., `\begin{matrix}`) are safely extracted and bypassed.
- **Newline Conversion**: R newline characters (`\n`) occurring outside of math environments are automatically converted to LaTeX line breaks (`\\`) inside the `\text{}` wrapper.
- **Literal Escapes**: Escaped LaTeX literals (e.g., `\$`, `\%`, `\#`) are safely passed into the `\text{}` block without triggering math modes. The escape character for `\$` is automatically resolved for MicroTex compatibility.

Value

A `character` vector of the same length as `tex`, formatted for math-mode LaTeX rendering.

Examples

```
# "mixed" mode (default) safely wraps text and preserves inline math
latex_wrap(r"(The equation \(E=mc^2\) is famous)")

# "mixed" mode handles user-escaped characters seamlessly
latex_wrap(r"(Cost: \$100 for $x$ items)")

# "mixed" mode converts R newlines to stacked text blocks
latex_wrap(r"(Line 1\nLine 2)")

# "math" mode returns the string completely unmodified
latex_wrap(r"(\frac{\alpha}{\beta})", input_mode = "math")
```

---

load_font *Load a math font from an OTF file*

Description

Loads an OTF/TTF math font into MicroTeX's internal font registry. The font's OpenType MATH table is parsed directly in C++ and the required metrics are synthesised on the fly. You can download free math fonts like Latin Modern Math (default math fonts in LaTeX) and load it with `load_font()` to use it for math rendering.

Usage

```
load_font(otf_path)
```

Arguments

`otf_path`	Path to the OTF/TTF font file.

Details

The font is also registered with the **systemfonts** package so it can be selected for surrounding plot text via `gp = gpar(fontfamily = "...")` without being installed system-wide.

Value

Invisibly returns `NULL`.

Text fonts

This function is only for **math** fonts (fonts with an OpenType MATH table). Plain text fonts used inside `\text{}` blocks are resolved automatically by **systemfonts** from the `gp$fontfamily` argument — no `load_font()` call required.

See Also

`available_math_fonts`, `latex_options`, `latex_grob`

Examples

```
# Load a math font from a local OTF file. Here we point at the
# bundled STIX font so the example is self-contained and loaded.
# You don't need to load the bundled fonts to use them — they're registered
# with systemfonts on first render — but this shows how to load a custom font.
# in practice you would pass the path to any OTF with an OpenType MATH table.
otf <- system.file("fonts", "STIXTwoMath-Regular.otf",
                   package = "gridmicrotex")
load_font(otf)
available_math_fonts()
```

Index