Package 'tikzDevice'

November 29, 2023

Encoding UTF-8

Type Package

Title R Graphics Output in LaTeX Format

Version 0.12.6

URL https://github.com/daqana/tikzDevice

BugReports https://github.com/daqana/tikzDevice/issues

Description Provides a graphics output device for R that records plots in a LaTeX-friendly format. The device transforms plotting commands issued by R functions into LaTeX code blocks. When included in a LaTeX document, these blocks are interpreted with the help of 'TikZ'---a graphics package for TeX and friends written by Till Tantau. Using the 'tikzDevice', the text of R plots can contain LaTeX commands such as mathematical formula. The device also allows arbitrary LaTeX code to be inserted into the output stream.

License GPL (>= 2)

Depends R (>= 2.14.0)

Imports filehash (>= 2.3), png

Suggests evaluate, formatR, ggplot2, knitr, lattice, maps, scales, stringr, testthat (>= 0.8.1), withr, covr

SystemRequirements pgf (>= 2.00)

LazyLoad yes

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation yes

Author Charlie Sharpsteen [aut],
Cameron Bracken [aut],
Kirill Müller [ctb],
Yihui Xie [ctb],
Ralf Stubner [cre],
Nico Bellack [ctb]

Maintainer Ralf Stubner <ralf.stubner@gmail.com>

Repository CRAN

Date/Publication 2023-11-29 21:40:02 UTC

R topics documented:

`tikzDevice-package` *Support for native LaTeX output of R graphics*

Description

The tikzDevice package implements the `tikz()` ouput device which generates R graphics in a LaTeX friendly format. LaTeX handles the typesetting of all text in graphics generated by `tikz`. This allows for seamless integration between these graphics and documents that are also being typeset by LaTeX. Using LaTeX to generate graph text also means that **LaTeX mathematics can be typeset directly into labels and annotations**.

Options That Affect Package Behavior

The **tikzDevice** package is currently influenced by a number of global options that may be set in scripts, from the console or in a `.Rprofile` file. All of the options can be set by using `options(<option> = <value>)`. These options allow for the use of custom `documentclass` declarations, LaTeX packages, and typesetting engines (e.g. XeLaTeX or LuaLaTeX). The defaults, if are any for a given option, are shown below the description. The global options are:

`tikzDefaultEngine` Specifies which typesetting engine functions in the tikzDevice package will prefer. Current possible values are `pdftex`, `xetex` or `luatex`. Respectively, these values trigger the use of the `pdflatex`, `xelatex` and `lualatex` compilers.

`tikzLatex` Specifies the location of the LaTeX compiler to be used by **tikzDevice**. Setting this option may help the package locate a missing compiler. The default is searched for when the package is loaded, otherwise it can be set manually. This option may be set as follows: `options( tikzLatex = '/path/to/latex/compiler' )`.

`tikzXelatex` Functions similar to `tikzLatex`, except this option specifies the location of the XeLaTeX compiler.

`tikzLualatex` Functions similar to `tikzLatex`, except this option specifies the location of the LuaLaTeX compiler.

`tikzMetricsDictionary` When using the graphics device provided by **tikzDevice**, you may notice that appears to "lag" or "hang" when commands such as `plot()` are executed. This is because the device must query the LaTeX compiler for string widths and font metrics. For a normal plot, this may happen dozens or hundreds of times- hence becomes unresponsive for a while. The good news is that the `tikz()` code is designed to cache the results of these computations so they need only be performed once for each string or character. By default, these values are stored in a temporary cache file which is deleted when is shut down. A location for a permanent cache file may be specified by setting the value of `tikzMetricsDictionary` in `.Rprofile` with `options(tikzMetricsDictionary = '/path/to/dictionary/location')`.

`tikzDocumentDeclaration` A string. The LaTeX `documentclass` declaration used in output files when `standAlone == TRUE`. `tikzDocumentDeclaration` also influences the calculation of font metrics. The default value is: `options(tikzDocumentDeclaration = "\\documentclass[10pt]{article}")`

`tikzLatexPackages` A character vector. These are the packages which are included when using the `pdftex` engine and `tikz()` is used with the the `standAlone` option as well as when font metrics are calculated.

`tikzXelatexPackages` This option works like `tikzLatexPackages`, except is is used when the `xetex` engine is in use.

`tikzLualatexPackages` This option works like `tikzXelatexPackages`, except is is used when the `luatex` engine is in use.

`tikzFooter` A character vector. The footer to be used only when `standAlone==TRUE`.

`tikzMetricPackages` A character vector. These are the packages which are additionally loaded when doing font metric calculations. As you see below, the font encoding is set to Type 1. This is very important so that character codes of LaTeX and match up. The default value is: `options(tikzMetricPackages = c( "\\usepackage[utf8]{inputenc}", "\\usepackage[T1]{fontenc}", "\\usetikzlibrary{calc}" ))`

`tikzUnicodeMetricPackages` This vector is used when font metric calculations are performed using the `xetex` or `luatex` engines. It should have the same contents as `tikzMetricPackages` with the addition of the `fontspec` and `xunicode` packages.

`tikzSanitizeCharacters` A character vector of special latex characters to replace. These values should correspond to the replacement values from the `tikzReplacementCharacters` option. See `sanitizeTexString()` for more details.

`tikzReplacementCharacters` A character vector of replacements for special latex characters. These values should correspond to the values from the `tikzSanitizeCharacters` option.

`tikzLwdUnit` A numeric that denotes the number of `pts` in LaTeX that `lwd=1` in R is translated to. Defaults to 0.4 (LaTeX and TikZ default); for compatibility with R default, please use 72.27/96 (96 pixels in R is 1 inch, which is 72.27 points in TeX).

`tikzPdftexWarnUTF` A `TRUE/FALSE` value that controls whether warnings are printed if Unicode characters are sent to a device using the `pdftex` engine.

`tikzSymbolicColors` A logical value indicating whether colors are written as RGB values or as symbolic names in which case the need to be defined in the LaTeX document.

`tikzMaxSymbolicColors` an integer number indicating the maximal number of distinct colors to write symbolically. Any excess color will be defined as if `symbolicColors` was set to `FALSE`.

Default values for all options may be viewed or restored using the `setTikzDefaults()` function.

Author(s)

See `packageDescription("tikzDevice")`.

Submit bug reports to: https://github.com/daqana/tikzDevice/issues

References

The TikZ and PGF Packages: Manual for version 2.00
https://sourceforge.net/projects/pgf
Till Tantau, February 20, 2008

See Also

`tikz()`

---

anyMultibyteUTF8Characters

Check If a String Contains Multibyte UTF-8 characters

Description

This function is used by tikzDevice to check if an incoming string contains multibyte UTF-8 characters

Usage

```
anyMultibyteUTF8Characters(string, encoding = "UTF-8")
```

Arguments

`string`	A character vector of length 1 (a string).
`encoding`	Unused.

Details

This function searches through the characters in the given string, if any of the characters in the string are more than one byte then the function returns `TRUE` otherwise it returns `FALSE`.

The function will assume an input encoding of UTF-8 but will take any specified encoding into account and will convert from the specified encoding to UTF-8 before doing any checks

Value

A boolean value

Author(s)

Cameron Bracken <cameron.bracken@gmail.com>

See Also

tikz()

Examples

```
# TRUE
anyMultibyteUTF8Characters('R is GNU ©, but not ®')
# FALSE
anyMultibyteUTF8Characters('R is GNU copyright but not restricted')
```

---

getLatexStrWidth *Obtain Font Metrics from LaTeX*

Description

These functions calculate the width of a character or string as it would appear after being compiled by LaTeX.

Usage

```
getLatexStrWidth(
  texString,
  cex = 1,
  face = 1,
  engine = getOption("tikzDefaultEngine"),
  documentDeclaration = getOption("tikzDocumentDeclaration"),
  packages,
  verbose = interactive(),
  diagnose = FALSE
)

getLatexCharMetrics(
  charCode,
  cex = 1,
  face = 1,
  engine = getOption("tikzDefaultEngine"),
  documentDeclaration = getOption("tikzDocumentDeclaration"),
  packages,
  verbose = interactive()
)
```

Arguments

texString	An arbitrary string for which the width is to be calculated. May contain LaTeX markup.
cex	a real number that specifies a scaling factor that is to be applied to device output.
face	an integer in the range 1:5 that specifies the font face to use. See par for details.
engine	a string specifying which TeX engine to use. Possible values are 'pdftex', 'xetex' and 'luatex'. See the Unicode section of tikzDevice-package for details.
documentDeclaration	See the sections "Options That Affect Package Behavior" and "Font Size Calculations" of tikzDevice-package for more details.
packages	See the section "Options That Affect Package Behavior" of tikzDevice-package.
verbose	A logical value indicating whether diagnostic messages are printed when measuring dimensions of strings. Defaults to TRUE in interactive mode only, to FALSE otherwise.
diagnose	pass TRUE to print detailed error information.
charCode	an integer that corresponds to a symbol in the ASCII character table under the Type 1 font encoding. All numeric values are coerced using as.integer(). Non-numeric values will not be accepted.

Details

These functions are used internally by the tikz device for proper string placement in graphics. Both functions check to see if metrics exist in a global or temporary dictionary (as defined in options('tikzMetricsDictionary')) and if so will pull the metrics from there. If the dictionary does not exist, then a temporary one is created for the current R session. Metrics are calculated via system calls to LaTeX compilers. Querying compilers to calculate metrics is expensive and so we strongly recommend setting options('tikzMetricsDictionary') <- '/path/to/dictionary' to create a global dictionary.

Value

getLatexStrWidth	The width of texString in points.
getLatexCharMetrics	A numeric vector holding ascent, descent and width. Values should all be non-negative.

Author(s)

Charlie Sharpsteen <source@sharpsteen.net> and Cameron Bracken <cameron.bracken@gmail.com>

References

PGF Manual

Examples

```
getLatexStrWidth('{\\\\tiny Hello \\\\LaTeX!}')


# Calculate ascent, descent and width for "A"
getLatexCharMetrics(65)
```

`gridToDevice` *Convert grid coordinates to device coordinates*

Description

This function converts a coordinate pair specifying a location in a grid `viewport()` in grid units to a coordinate pair specifying a location in device units relative to the lower left corner of the plotting canvas.

Usage

```
gridToDevice(x = 0, y = 0, units = "native")
```

Arguments

`x`	x coordinate.
`y`	y coordinate. If no values are given for `x` and `y`, the location of the lower-left corner of the current viewport will be calculated.
`units`	Character string indicating the units of `x` and `y`. See the `unit()` function for acceptable unit types.

Value

A tuple of coordinates in device units.

Author(s)

Charlie Sharpsteen <source@sharpsteen.net>

See Also

`unit()` `viewport()` `convertX()` `convertY()` `current.transform()`

`sanitizeTexString` *Replace LaTeX Special Characters in a String*

Description

This function is used by tikzDevice when `sanitize = TRUE` to replace special LaTeX characters (such as the comment character %) in plotting text where the user does not have direct control over the generated text.

Usage

```
sanitizeTexString(
  string,
  strip = getOption("tikzSanitizeCharacters"),
  replacement = getOption("tikzReplacementCharacters")
)
```

Arguments

`string`	A character vector of length 1 (a string).
`strip`	A character vector of single characters to search for.
`replacement`	A character vector of replacement values.

Details

`sanitizeTexString()` searches character by character through a string replacing each occurrence of a special character contained in `strip[i]` with the corresponding replacement value in `replacement[i]`. tikzDevice calls back this function for every piece of text when the sanitize option is TRUE. See `tikz()` for more information on the default special characters and replacement values.

By default, `tikzSanitizeCharacters` replaces the following characters:

- %
- $
- }
- {
- ^
- _
- #
- &
- ~

With the contents of `tikzReplacementCharacters`:

- \%
- \$

- \}
- \{
- \^{}
- _{}
- \#
- \&
- \char`\~

These defaults may be adjusted using the options() function.

Value

sanitizedString
: A character vector of length 1 with all special characters replaced.

Author(s)

Cameron Bracken <cameron.bracken@gmail.com>

See Also

tikz()

Examples

```
# Be careful with sanitizing, it may lead to unexpected behavior.
# For example, we may want -1 to be a superscript it gets
# sanitized away with the other default special characters.
# The string appears in LaTeX exactly as shown.
## Not run:
  sanitizeTexString('10\

## End(Not run)
```

---

setTikzDefaults *Reset tikzDevice options to default values.*

Description

This function resets the following options:

Usage

```
setTikzDefaults(overwrite = TRUE)
```

Arguments

overwrite	Should values that are allready set in options() be overwritten?

Details

- tikzDefaultEngine
- tikzLatex
- tikzDocumentDeclaration
- tikzFooter
- tikzLatexPackages
- tikzXelatexPackages
- tikzLualatexPackages
- tikzMetricPackages
- tikzUnicodeMetricPackages
- tikzSanitizeCharacters
- tikzReplacementCharacters
- tikzPdftexWarnUTF

Value

Nothing returned.

Author(s)

Cameron Bracken <cameron.bracken@gmail.com> and Charlie Sharpsteen <source@sharpsteen.net>

See Also

tikz()

Examples

```
print( options( 'tikzDocumentDeclaration' ) )
options( tikzDocumentDeclaration = 'foo' )
setTikzDefaults()
print( options( 'tikzDocumentDeclaration' ) )
```

tikz *TikZ Graphics Device*

Description

`tikz()` is used to open a R graphics device which supports output in the TikZ graphics language. TikZ code may be included inside a LaTeX document by specifying `\usepackage{tikz}` in the document header.

Usage

```
tikz(
  file = filename,
  filename = ifelse(onefile, "./Rplots.tex", "./Rplot%03d.tex"),
  width = 7,
  height = 7,
  onefile = TRUE,
  bg = "transparent",
  fg = "black",
  pointsize = 10,
  lwdUnit = getOption("tikzLwdUnit"),
  standAlone = FALSE,
  bareBones = FALSE,
  console = FALSE,
  sanitize = FALSE,
  engine = getOption("tikzDefaultEngine"),
  documentDeclaration = getOption("tikzDocumentDeclaration"),
  packages,
  footer = getOption("tikzFooter"),
  symbolicColors = getOption("tikzSymbolicColors"),
  colorFileName = "%s_colors.tex",
  maxSymbolicColors = getOption("tikzMaxSymbolicColors"),
  timestamp = TRUE,
  verbose = interactive()
)
```

Arguments

`file, filename`	A character string indicating the desired path to the output file. If both arguments are used in the function call, `file` will be preferred.
`width`	The width of the output figure, in **inches**.
`height`	The height of the output figure, in **inches**.
`onefile`	Should output be directed to separate environments in a single file (default `TRUE`). If `FALSE` this option works exactly like the argument of the same name to `pdf()` (see there for more details).
`bg`	The starting background color for the plot.

`fg` The starting foreground color for the plot.

`pointsize` Base pointsize used in the LaTeX document. This option is only used if a valid pointsize cannot be extracted from the value of `getOption("tikzDocumentDeclaration")`. See the section "Font Size Calculations" in tikzDevice-package for more details.

`lwdUnit` The number of `pt`s in LaTeX that `lwd = 1` in R is translated to. Defaults to 0.4 (LaTeX and TikZ default); for compatibility with R default, please use 72.27/96 (96 pixels in R is 1 inch, which is 72.27 points in TeX).

`standAlone` A logical value indicating whether the output file should be suitable for direct processing by LaTeX. A value of `FALSE` indicates that the file is intended for inclusion in a larger document. See 'Details'.

`bareBones` A logical value. When `TRUE` the figure will not be wrapped in a `tikzpicture` environment. This option is useful for embedding one TikZ picture within another. When `TRUE` multipage output will be drawn on a single page.

`console` Should the output of tikzDevice be directed to the R console (default `FALSE`). This is useful for dumping tikz output directly into a LaTeX document via `sink()`. If TRUE, the `file` argument is ignored. Setting `file = ''` is equivalent to setting `console=TRUE`.

`sanitize` Should special latex characters be replaced (Default FALSE). See the section "Options That Affect Package Behavior" for which characters are replaced.

`engine` a string specifying which TeX engine to use. Possible values are 'pdftex', 'xetex' and 'luatex'. See the Unicode section of tikzDevice-package for details.

`documentDeclaration`
See the sections "Options That Affect Package Behavior" and "Font Size Calculations" of tikzDevice-package for more details.

`packages` See the section "Options That Affect Package Behavior" of tikzDevice-package.

`footer` See the section "Options That Affect Package Behavior" of tikzDevice-package.

`symbolicColors` A logical value indicating whether colors are written as RGB values or as symbolic names in which case the need to be defined in the LaTeX document. These definitions can be generated with the following `colorFileName` parameter. See also the section "Options That Affect Package Behavior" of tikzDevice-package.

`colorFileName` a character string indicating where the color map for symbolic colors is to be stored. It can contain a placeholder `%s` where the tikz filename is inserted. If the string is empty, no file is written.

`maxSymbolicColors`
an integer number indicating the maximal number of distinct colors to write symbolically. Any excess color will be defined as if `symbolicColors` was set to `FALSE`. See also the section "Options That Affect Package Behavior" of tikzDevice-package.

`timestamp` A logical value indicating whether a timestamp is written to the TeX file.

`verbose` A logical value indicating whether diagnostic messages are printed when measuring dimensions of strings. Defaults to `TRUE` in interactive mode only, to `FALSE` otherwise.

Details

The TikZ device enables LaTeX-ready output from graphics functions. This is done by encoding graphics commands using TikZ markup. All text in a graphic output with `tikz` will be typeset by LaTeX and therefore will match whatever fonts are currently used in the document. This also means that **LaTeX mathematics can be typeset directly into labels and annotations**.

The TikZ device currently supports three modes of output depending on the value of the `standAlone` and `bareBones` arguments. If `standAlone` and `bareBones` are set to the default value of `FALSE`, the resulting file will only contain graphics output wrapped in a LaTeX `tikzpicture` environment. Since this file is not a complete LaTeX document, it will need to be included in another LaTeX document using the `\\input` command. For example:

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\begin{figure}
  \centering
  \input{Rplots.tex}
  \caption{}
\end{figure}
\end{document}
```

When `standAlone` is set to `TRUE`, the device wraps the `tikzpicture` environment in a complete LaTeX document suitable for direct compilation. In this mode the `preview` package is used to crop the resulting output to the bounding box of the graphic.

When `bareBones` is set to `TRUE`, the output is not wrapped in a document or a `tikzpicture` environment. This is useful for embedding an generated graphic within an existing TikZ picture.

In cases where both `standAlone` and `bareBones` have been set to `TRUE`, the `standAlone` option will take precedence.

When the option `symbolicColors` is set to `TRUE`, the colors will be written as symbolic names, e.g. `red, gray90` and similar. If the color is not mapped to a symbolic name in R, the color will be named `XXXXX` when `#XXXXXX` is its hexadecimal color. All the color names will have to be defined in the enclosing document, which is automatically written if the path of a color file `colorFileName` is set.

Value

`tikz()` returns no values.

Note

To compile the output of `tikz` a working installation of LaTeX and PGF is needed. Current releases of the TikZ package are available from https://www.ctan.org. The package may also be installed through the MikTeX package manager on Windows or using the TeX Live package manager, `tlmgr`, on Unix/Linux/OS X. The TeX Live package manager will only be installed by default for TeX Live distributions dated 2008 and later. Both bleeding-edge and release versions of TikZ may be obtained from the project website hosted at https://sourceforge.net/projects/pgf/.

Multiple plots will be placed as separate environments in the output file.

Author(s)

Charlie Sharpsteen <source@sharpsteen.net> and Cameron Bracken <cameron.bracken@gmail.com>

References

The TikZ and PGF Packages: Manual for version 2.00
https://sourceforge.net/projects/pgf
Till Tantau, February 20, 2008

See Also

pictex(), getLatexCharMetrics(), getLatexStrWidth(), setTikzDefaults(), tikzAnnotate(), sanitizeTexString()

Examples

```
## Not run:

## Example 1 ###################################
#Set up temporary work directory
td <- tempdir()
tf <- file.path(td,'example1.tex')
oldwd <- getwd()
setwd(td)

# Minimal plot
tikz(tf,standAlone=TRUE)
  plot(1)
dev.off()

# View the output
tools::texi2dvi(tf,pdf=T)
system(paste(getOption('pdfviewer'),file.path(td,'example1.pdf')))
setwd(oldwd)
##############################################

## Example 2 ###################################
#Set up temporary work directory
td <- tempdir()
tf <- file.path(td,'example2.tex')
oldwd <- getwd()
setwd(td)

#LaTeX math symbol names
syms <-c('alpha','theta','tau','beta','vartheta','pi','upsilon',
         'gamma','gamma','varpi','phi','delta','kappa','rho',
         'varphi','epsilon','lambda','varrho','chi','varepsilon',
         'mu','sigma','psi','zeta','nu','varsigma','omega','eta',
         'xi','Gamma','Lambda','Sigma','Psi','Delta','Xi','Upsilon',
         'Omega','Theta','Pi','Phi')
x <- rnorm(length(syms))
```

```
y <- rnorm(length(syms))

tikz(tf,standAlone=TRUE)
  plot(-2:2, -2:2, type = "n", axes=F,
      xlab='', ylab='', main='TikZ Device Math Example')
    text(x,y,paste('\\\\Large$\\\\',syms,'$',sep=''))
dev.off()

#View the output
tools::texi2dvi(tf,pdf=TRUE)
system(paste(getOption('pdfviewer'),file.path(td,'example2.pdf')))
setwd(oldwd)
################################################

## Example 3 ###################################
#Set up temporary work directory
td <- tempdir()
tf <- file.path(td,'example3.tex')
oldwd <- getwd()
setwd(td)

tikz(tf,standAlone=TRUE)
  plot(-2:2, -2:2, type = "n", axes=F, xlab='', ylab='', main='Random Circles')
    points(rnorm(50), rnorm(50), pch=21,
      bg=rainbow(50,alpha=.5), cex=10)
dev.off()

#View the output
tools::texi2dvi(tf,pdf=TRUE)
system(paste(getOption('pdfviewer'),file.path(td,'example3.pdf')))
setwd(oldwd)
################################################

## End(Not run)
```

`tikzAnnotate` *Add Custom TikZ Code to an Active Device*

Description

These functions allow custom (LaTeX) commands to be added to the output of an active tikzDevice.

Usage

```
tikzAnnotate(annotation, checkstate = TRUE)

tikzNode(
  x = NULL,
  y = NULL,
```

```
  opts = NULL,
  name = NULL,
  content = NULL,
  units = "user"
)

tikzCoord(x, y, name, units = "user")

tikzAnnotateGrob(annotation)

tikzNodeGrob(
  x = NULL,
  y = NULL,
  opts = NULL,
  name = NULL,
  content = NULL,
  units = "native"
)

tikzCoordGrob(x, y, name, units = "native")

grid.tikzAnnotate(annotation, draw = TRUE)

grid.tikzNode(
  x = NULL,
  y = NULL,
  opts = NULL,
  name = NULL,
  content = NULL,
  units = "native",
  draw = TRUE
)

grid.tikzCoord(x, y, name, units = "native", draw = TRUE)
```

Arguments

annotation	A character vector, one element per line to be added to the open tikz device.
checkstate	A logical, whether to "flush" the device state prior to writing the annotation.
x	numeric, x location for a named coordinate in user coordinates
y	numeric, y location for a named coordinate in user coordinates
opts	A character string that will be used as options for a node. See the "Nodes and Edges" section of the TikZ manual for complete details.
name	Optional character string that will be used as a name for a coordinate or node. Other TikZ commands can use this name to refer to a location in a graphic.
content	A character string that will be used as the content to be displayed inside of a node. If left as NULL a coordinate will be created instead of a node. If a node with empty content is truely desired, pass an empty string "".

units	Character string specifying the unit system associated with x and y. See grconvertX() for acceptable units in base graphics and unit() for acceptable units in grid graphics.
draw	A logical value indicating whether graphics output should be produced.

Details

tikzAnnotate is intended to allow the insertion of arbitrary TikZ commands into the output stream of a graphic. For LaTeX commands that reference specific locations in an R plot, coordinates must be specified in "device units" which for tikz output are TeX points relative to the lower left corner of the device canvas. Functions such as grconvertX() and gridToDevice() can help make the necessary conversions for base and grid graphics. The tikzNode and tikzCoord functions automatically perform unit conversions acording the the value of their units parameters.

tikzNode is a wrapper for tikzAnnotate that inserts TikZ \\node or \\coordinates commands into the output. The difference between a node and a coordinate is the presence of a content section that can contain arbitrary LaTeX text. This is useful for adding textual annotations at specific locations in a TikZ graphic. The tikzCoord function is a wrapper for tikzNode that simplifies the task of inserting named coordinates.

Additionally, the tikzAnnotateGrob, tikzNodeGrob and tikzCoordGrob functions are supplied for creating grid objects or "grob()s" that can be used in Grid graphics. High level wrapper functions grid.tikzAnnotate, grid.tikzNode and grid.tikzCoord are also supplied which creat and render a grob in one step.

See the TikZ Device vignette for more information and examples and the TikZ manual for the definitive reference on what is possible with nodes.

Value

Nothing returned.

Author(s)

Cameron Bracken cameron.bracken@gmail.com and Charlie Sharpsteen <source@sharpsteen.net>

See Also

grconvertX() grconvertY() gridToDevice() unit() tikz()

Examples

```
## Not run:

### Example 1: Annotations in Base Graphics
# Load some additional TikZ libraries
tikz("annotation.tex",width=4,height=4,
  packages = c(getOption('tikzLatexPackages'),
    "\\usetikzlibrary{decorations.pathreplacing}",
    "\\usetikzlibrary{positioning}",
    "\\usetikzlibrary{shapes.arrows,shapes.symbols}")
```

```
)

p <- rgamma (300 ,1)
outliers <- which( p > quantile(p,.75)+1.5*IQR(p) )
boxplot(p)

# Add named coordinates that other TikZ commands can hook onto
tikzCoord(1, min(p[outliers]), 'min outlier')
tikzCoord(1, max(p[outliers]), 'max outlier')

# Use tikzAnnotate to insert arbitrary code, such as drawing a
# fancy path between min outlier and max outlier.
tikzAnnotate(c("\\draw[very thick,red,",
  # Turn the path into a brace.
  'decorate,decoration={brace,amplitude=12pt},',
  # Shift it 1em to the left of the coordinates
  'transform canvas={xshift=-1em}]',
  '(min outlier) --',
  # Add a node with some text in the middle of the path
  'node[single arrow,anchor=tip,fill=white,draw=green,',
  'left=14pt,text width=0.70in,align=center]',
  '{Holy Outliers Batman!}', '(max outlier);'))

# tikzNode can be used to place nodes with customized options and content
tikzNode(
  opts='starburst,fill=green,draw=blue,very thick,right=of max outlier',
  content='Wow!'
)

dev.off()


### Example 2: Annotations in Grid Graphics
library(grid)

tikz("grid_annotation.tex",width=4,height=4,
  packages = c(getOption('tikzLatexPackages'),
    "\\usetikzlibrary{shapes.callouts}")
)

pushViewport(plotViewport())
pushViewport(dataViewport(1:10, 1:10))

grid.rect()
grid.xaxis()
grid.yaxis()
grid.points(1:10, 1:10)

for ( i in seq(2,8,2) ){
  grid.tikzNode(i,i,opts='ellipse callout,draw,anchor=pointer',content=i)
}

dev.off()
```

```
## End(Not run)
```

`tikzCompilerInfo` *Print paths to TeX compilers.*

Description

This function reports information concerning compilers that the `tikz` device will use to calculate character metrics. Information on LaTeX will always be available but information on XeLaTeX and LuaLaTeX will only be reported if the compilers were found.

Usage

```
tikzCompilerInfo(verbose = TRUE)
```

Arguments

`verbose`	If set to `FALSE`, calling this function will not cause any output to be printed to the screen. Defaults to `TRUE`.

Value

Invisibly returns a list containing paths to TeX compilers.

Author(s)

Charlie Sharpsteen <source@sharpsteen.net>

See Also

`tikz()`

tikzTest *Test invocation of a LaTeX engine.*

Description

This function simulates the measurement of dimensions and prints detailed information in case of errors.

Usage

```
tikzTest(
  texString = "A",
  engine = getOption("tikzDefaultEngine"),
  documentDeclaration = getOption("tikzDocumentDeclaration"),
  packages
)
```

Arguments

texString	An arbitrary string for which the width is to be calculated. May contain LaTeX markup.
engine	a string specifying which TeX engine to use. Possible values are 'pdftex', 'xetex' and 'luatex'. See the Unicode section of tikzDevice-package for details.
documentDeclaration	See the sections "Options That Affect Package Behavior" and "Font Size Calculations" of tikzDevice-package for more details.
packages	See the section "Options That Affect Package Behavior" of tikzDevice-package.

See Also

tikz()

Index