

Package ‘tikzDevice’

January 13, 2012

Type Package

Title A Device for R Graphics Output in PGF/TikZ Format

Version 0.6.2

Date 2011-11-13

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

Maintainer The tikzDevice team <tikzdevice-bugs@lists.r-forge.r-project.org>

URL <https://github.com/Sharpie/RTikZDevice>

BugReports <https://github.com/Sharpie/RTikZDevice/issues>

Description The TikZ device enables LaTeX-ready output from R graphics functions. This is done by producing code that can be understood by the TikZ graphics language. All text in a graphic output with the tikz() function will can 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! Graphics produced this way can also be annotated with custom TikZ commands.

License GPL (>= 3)

Depends R (>= 2.11.0), filehash

Suggests testthat (>= 0.4), evaluate, stringr, ggplot2, maps

SystemRequirements pgf (>= 2.00)

LazyLoad yes

Repository CRAN

Date/Publication 2012-01-13 21:35:54

R topics documented:

tikzDevice-package	2
anyMultibyteUTF8Characters	5
getLatexCharMetrics	6
getLatexStrWidth	8
gridToDevice	9
sanitizeTexString	10
setTikzDefaults	11
tikz	12
tikzAnnotate	16

Index	19
--------------	-----------

tikzDevice-package	<i>Support for native LaTeX output of R graphics</i>
--------------------	--

Description

The `tikzDevice` package implements the `tikz` output 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**

Details

Package:	tikzDevice
Type:	Package
Version:	0.6.2
Date:	2011-11-13
License:	GPL 3.0 or greater
LazyLoad:	yes

Options That Affect Package Behavior

The `tikzDevice` package is currently influenced by a number of global options that may be set in R scripts, from the R 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 LuaTeX). The defaults, if 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` or `xetex` which will respectively trigger the use of the `pdflatex` and `xelatex` 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 Xe-LaTeX compiler.

`tikzMetricsDictionary` When using the graphics device provided by **tikzDevice**, you may notice that R 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 R 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 R 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 `standAlone` option as well as when font metrics are calculated.

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

`tikzRasterResolution` When `tikz` is requested to add a raster to a graphic, the raster is written to a PNG file which is then included into the LaTeX code. This option controls the resolution (dpi) at which the PNG files are created.

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

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

Font Size Calculations

The overarching goal of the `tikzDevice` is to provide seamless integration between text in R graphics and the text of LaTeX documents that contain those graphics. In order to achieve this integration the device must translate font sizes specified in R to corresponding font sizes in LaTeX.

A major consideration is that font sizes in LaTeX are controlled by a “base font size” that is specified at the beginning of the document; typically 10pt. Furthermore, LaTeX typically provides only a finite number of sizes for each font. Fortunately, the TikZ graphics system allows text to be resized using a scaling factor. The `tikzDevice` calculates this scaling factor using three inputs:

- The “base font size” specified when the graphics device is created.
- The “character expansion factor” parameter, specified using the `cex` argument to functions such as `par`.
- The “font size” parameter, specified using the `ps` argument to functions such as `par` or the `fontsize` argument to functions such as `gpar`.

The tricky bit is the specification of the “base font size”. By default the `tikzDevice` will attempt to determine this parameter by scanning the value of `options("tikzDocumentDeclaration")` using the regular expression `\d+[pt]`. For the default header of:

```
\documentclass[10pt]{article}
```

This regular expression will return 10 as the base pointsize to be used by the device. If the regular expression fails to produce a match, the value of the `pointsize` argument to the `tikz` function will be used.

Unicode

Using `tikzDevice` functions on strings containing Unicode (multibyte) characters requires special consideration. The primary reason for this is that pdfTeX, the typesetting engine most commonly used to compile LaTeX documents, has no native support for Unicode. Recently, two new engines have become widely available that do support Unicode natively. These engines are:

- The `xetex` engine which has been a part of standard TeX distributions since TeX Live 2007 and MiKTeX 2.7.
- The `luatex` engine which is a direct successor to pdfTeX. LuaTeX is still considered ‘beta’ software (as of spring 2011) but has been included in standard TeX distributions since TeX Live 2008 and MiKTeX 2.9.

As of Version 0.6.0, the tikzDevice package includes support for the xetex engine provided a recent version of TeX Live or MiKTeX is available.

The tikzDevice defaults to using the pdftex engine, but this may be changed by setting an option: `option(tikzDefaultEngine = 'xetex')`. When working with text that contains Unicode characters, we strongly recommend switching to the xetex engine and the package will produce warning messages when multibyte characters are sent to the pdftex engine. Users who find these warnings annoying may disable them by setting `options(tikzPdfTeXWarnUTF = FALSE)` as it is possible to send Unicode information to the pdftex engine with the following caveats:

- The task of determining the proper mix of LaTeX packages required to supply glyphs for Unicode characters is left entirely up to the user. `options(tikzLatexPackages)` may need to be customized in order to get proper behavior.
- The `getLatexCharMetrics` function will always return 0 for ascent, descent and width when passed a character code outside the range 32–127 when using the pdftex engine.

When using the xetex engine, Unicode support is enabled by loading the LaTeX packages `fontspec` and `xunicode`. Using this setup, support for Unicode characters should be limited only by the fonts used in the document. The font selection may be adjusted using `options(tikzXeLaTeXPackages)`.

Author(s)

Cameron Bracken: <cameron.bracken@gmail.com>

Charlie Sharpsteen: <source@sharpsteen.net>

Submit bug reports to: <tikzdevice-bugs@lists.r-forge.r-project.org>

References

The TikZ and PGF Packages: Manual for version 2.00

<http://sourceforge.net/projects/pgf>

Till Tantau, February 20, 2008

See Also

[tikz](#)

anyMultibyteUTF8Characters

Check If a String Contains Multibyte UTF-8 characters...

Description

Check If a String Contains Multibyte UTF-8 characters 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	The input encoding of string, if not specified previously via Encoding or by this argument then a value of "UTF-8" is assumed

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')
```

getLatexCharMetrics *Obtain LaTeX Font Metrics for Characters...*

Description

Obtain LaTeX Font Metrics for Characters This function is used to retrieve the ascent, decent and width of a character glyph as it would appear in output typeset by LaTeX.

Usage

```
getLatexCharMetrics(charCode, cex=1, face=1, engine=getOption("tikzDefaultEngine"))
```

Arguments

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 <code>as.integer</code> . Non-numeric values will not be accepted.
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 'pdf <code>tex</code> ' and 'x <code>etex</code> '. See the Unicode section of tikzDevice for details.

Details

`getLatexCharMetrics` first checks to see if metrics have already been calculated for the given character using the given values of `cex` and `face`. If so, cached values are returned. If no cached values exists, the LaTeX compiler specified by `options(tikzLatex)` is invoked in order to calculate them.

Value

metrics	A numeric vector holding ascent, descent, width character metrics. Values should all be nonnegative.
---------	--

Note

[tikzDevice](#) tries very hard when it is loaded to find a working `latex` or `pdflatex` command. If it is successful the command is set in `options('tikzLatex')`, otherwise this function will fail.

Author(s)

Charlie Sharpsteen <source@sharpsteen.net>

References

PGF Manual

See Also

[tikz](#), [getLatexStrWidth](#)

Examples

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

getLatexStrWidth *Obtain the Width of an Arbitrary LaTeX String...*

Description

Obtain the Width of an Arbitrary LaTeX String This function calculates the width of a string as it would appear after being compiled by LaTeX.

Usage

```
getLatexStrWidth(texString, cex=1, face=1, engine=getOption("tikzDefaultEngine"))
```

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' and 'xetex'. See the Unicode section of tikzDevice for details.

Details

This function is used internally by the tikz device for proper string placement in graphics. This function first checks to see if the width exists in a global or temporary string width dictionary (as define in options('tikzMetricsDictionary')) and if so will pull the width from there. If the dictionary does not exist then a temporary one for the current R session is created and the string width is calculated via a system call to latex. The calling of latex to calculate a string width is quit expensive and so we strongly recommend setting options('tikzMetricsDictionary') <- /path/to/dictionary to create a global dictionary.

Value

width	The width of texString in point size.
-------	---------------------------------------

Note

[tikzDevice](#) tries very hard when it is loaded to find a working latex or pdflatex command. If it is successful the command is set in options('tikzLatex'), otherwise this function will fail.

Author(s)

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

See Also

[tikz](#), [getLatexCharMetrics](#)

Examples

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

gridToDevice	<i>Convert grid coordinates to device coordinates...</i>
--------------	--

Description

Convert grid coordinates to device coordinates

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.

Details

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.

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

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

Usage

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

Arguments

<code>string</code>	A character vector of length 1 (a string).
<code>strip</code>	A character vector of single characters to search for.
<code>replacement</code>	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% of 10$ is 10^{-1}$')

## End(Not run)
```

<code>setTikzDefaults</code>	<i>Reset tikzDevice options.</i>
------------------------------	----------------------------------

Description

Reset tikzDevice options. Reset all the **tikzDevice** options to their default values.

Usage

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

Arguments

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

Details

Specifically resets the options `tikzLatex`, `tikzDocumentDeclaration`, `tikzLatexPackages`, `tikzMetricPackages`, `tikzFooter`, `tikzSanitizeCharacters` and `tikzReplacementCharacters`.

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 Graphics Device `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="./Rplots.tex", width=7, height=7, bg="transparent", fg="black",
      pointsize=10, standAlone=FALSE, bareBones=FALSE, console=FALSE,
      sanitize=FALSE, engine=getOption("tikzDefaultEngine"),
      documentDeclaration=getOption("tikzDocumentDeclaration"), packages,
      footer=getOption("tikzFooter"))
```

Arguments

<code>file</code>	A character string indicating the desired path to the output file.
<code>width</code>	The width of the output figure, in inches .
<code>height</code>	The height of the output figure, in inches .
<code>bg</code>	The starting background color for the plot.
<code>fg</code>	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 <code>getOption("tikzDocumentDeclaration")</code> . See the section “Font Size Calculations” in tikzDevice-package for more details.
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 <code>tikzpicture</code> 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 <code>tikzDevice</code> 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 <code>file</code> argument is ignored. Setting <code>file=""</code> is equivalent to setting <code>console=TRUE</code> .
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 ‘ <code>pdftex</code> ’ and ‘ <code>xetex</code> ’. 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 .

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 parameter `standAlone` and `bareBones`. 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.

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 <http://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 <http://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
<http://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

Add Custom TikZ Code to an Active Device

Usage

```

tikzAnnotate(annotation)
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.
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 truly 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

This function allows custom (LaTeX) commands to be added to the output of an active tikzDevice. 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 according to 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 "grobs" that can be used in Grid graphics. High level wrapper functions grid.tikzAnnotate, grid.tikzNode and grid.tikzCoord are also supplied which create 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
require(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)

```

Index

- *Topic **annotation**
 - tikzAnnotate, 16
 - *Topic **character**
 - anyMultibyteUTF8Characters, 5
 - getLatexCharMetrics, 6
 - getLatexStrWidth, 8
 - sanitizeTexString, 10
 - *Topic **conversion**
 - gridToDevice, 9
 - *Topic **device**
 - tikz, 12
 - tikzAnnotate, 16
 - *Topic **graphics**
 - gridToDevice, 9
 - *Topic **grid**
 - gridToDevice, 9
 - *Topic **package**
 - tikzDevice-package, 2
 - *Topic **tikz**
 - tikzAnnotate, 16
 - *Topic **units**
 - gridToDevice, 9
- anyMultibyteUTF8Characters, 5
- convertX, 9
- convertY, 9
- current.transform, 9
- Encoding, 6
- getLatexCharMetrics, 5, 6, 8, 14
- getLatexStrWidth, 7, 8, 14
- gpar, 4
- grconvertX, 16, 17
- grconvertY, 17
- grid.tikzAnnotate (tikzAnnotate), 16
- grid.tikzCoord (tikzAnnotate), 16
- grid.tikzNode (tikzAnnotate), 16
- gridToDevice, 9, 17
- grob, 17
- options, 11
- par, 4, 7, 8
- pictex, 14
- sanitizeTexString, 3, 10, 14
- setTikzDefaults, 4, 11, 14
- sink, 13
- tikz, 2–8, 10, 11, 12, 12, 17
- tikzAnnotate, 14, 16
- tikzAnnotateGrob (tikzAnnotate), 16
- tikzCoord (tikzAnnotate), 16
- tikzCoordGrob (tikzAnnotate), 16
- tikzDevice, 7, 8
- tikzDevice (tikzDevice-package), 2
- tikzDevice-package, 2, 13
- tikzNode (tikzAnnotate), 16
- tikzNodeGrob (tikzAnnotate), 16
- unit, 9, 16, 17
- viewport, 9