Package: plantuml (via r-universe)

September 17, 2024

Type Package

Title Create UML Graphs using PlantUML Syntax

Version 0.6.9

Date 2023-01-31

Maintainer Rainer M. Krug <Rainer@krugs.de>

Description Create UML graphs using the PlantUML language. These are either shown in a graphics device or saved as a file.

URL https://github.com/rkrug/plantuml

BugReports https://github.com/rkrug/plantuml/issues

License GPL-3

Encoding UTF-8

Depends R (> 4.0)

Imports utils, tools, rsvg, grDevices, graphics, jsonlite, png

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, covr, vdiffr, testthat, yaml

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/rkrug/plantuml

RemoteRef master

RemoteSha 34f33408c14925219a7acbf5338b9d35ddac2ab4

Contents

`addAttributes` *Add attributes to* `puml$code`

Description

Internal Only

Usage

```
addAttributes(x, nm, puml)
```

Arguments

`x`	the object
`nm`	the name
`puml`	the plantuml object

Value

puml

addInfo *Add info to* puml$code

Description

Internal Only

Usage

```
addInfo(x, nm, puml)
```

Arguments

x	the object
nm	the name
puml	the plantuml object

Value

puml

as.plantuml *Convert an R object to a Class uml object*

Description

Generic function to convert an R object to a plantuml Class object. See details for the individual functions

Usage

```
as.plantuml(x, ...)

## S3 method for class 'character'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'complex'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'data.frame'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## Default S3 method:
as.plantuml(x, complete = FALSE, nm = NULL, ...)
```

```
## S3 method for class 'factor'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'integer'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'list'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'logical'
as.plantuml(x, complete = FALSE, nm = NULL, ...)

## S3 method for class 'numeric'
as.plantuml(x, complete = FALSE, nm = NULL, ...)
```

Arguments

x	R object to be converted
...	other argunents passed on to generic functions

Value

object of class `plantuml` which can be plotted.

Examples

```
## Not run:
x <- list(
  a = 1:10,
  b = letters[1:4],
  c = data.frame(
    x = 1:10,
    y = c(TRUE, FALSE)
  )
)
x <- plantuml(x)
plot( x )

## End(Not run)

## Not run:
 x <- LETTERS
 x <- as.plantuml(x)
 plot(x)

## End(Not run)

## Not run:
 x <- factor(LETTERS)
 x <- as.plantuml(x)
 plot(x)
```

```
## End(Not run)

## Not run:
 x <- 1L:10L
 names(x) <- letters[1:10]
 x <- as.plantuml(x)
 plot(x)

## End(Not run)

## Not run:
 x <- list(
   num = 1:10,
   let = letters[1:20]
 )
 x$list <- list(x, x)
 x <- as.plantuml(x)
 plot(x)

## End(Not run)

## Not run:
 x <- 1/(-10:10)
 x <- as.plantuml(x)
 plot(x)

 x <- matrix(1:21, nrow = 7, ncol = 3)
 x <- as.plantuml(x)
 plot(x)

## End(Not run)
```

`diff_yaml_json` *Create diff between two lists to be used in the 'plantuml_yaml()' and 'plantuml_json()' as preamble to highlight the differences.*

Description

This function requires that the names of 'x' and 'y' are identical and in the same order. Any deviation from this will very likely result in strange error messages. The format follows the **0Highlight parts** on https://plantuml.com/yaml.

Usage

```
diff_yaml_json(x, y)
```

Arguments

x	list basde on yaml or json file to be compared. See the details zection for further info.
y	list based on yaml or json file to be compared. See the details zection for further info.

Value

'character' vector to be used as preamble for the 'plot_yaml()' and 'plantuml_json()' functions (or 'plantuml_file()' in case of a f=json or yaml file..

---

getPlantumlOption *Getter and Setter for options*

Description

Allow the user to set and examine the parameter used by the package `plantuml`

Usage

```
getPlantumlOption(x, default = NULL)

plantumlOptions(...)
```

Arguments

x	a character string holding an option name.
default	if the specified option is not set in the options list, this value is returned. This facilitates retrieving an option and checking whether it is set and setting it separately if not.
...	named value to be set in the form of `name = value`

Value

- `getPlantumlOption()`: the value of the option
- `plantumlOptions()`: the old value of the plantuml options

Examples

```
getPlantumlOption("jar_name")

plantumlOptions(jar_name = "something useless!")

getPlantumlOption("jar_name")
```

get_graph *Get PlantUML graph*

Description

Get the graph from either the local PicoWeb Server included in plantuml or an online plantuml server.

Usage

```
get_graph(x, file = NULL, width = NULL, height = NULL, css = NULL)
```

Arguments

x	plantuml code to draw the UML graph
file	file name, including extension, to which the returned plantUML graph should be saved. If NULL', the graph is saved to a temporary file. The following extensions are allow svg = tmpfile, file = file, width = width, height = height, css = css)- **pdf** Converted from **svg** u svg = tmpfile, file = file, width = width, height = height, css = css)- **ps** Converted from **svg** u svg = tmpfile, file = file, width = width, height = height, css = css)‘
width	output width in pixels or NULL for default.
height	output height in pixels or NULL for default
css	path/url to external css file or raw vector with css data. This requires your system has a recent version of librsvg.

Value

name of the file with the graph

plantuml *Convert a character to a* plantuml *object*

Description

Convert a character to a plantuml object. This can be plotted.

Usage

```
plantuml(x = NULL)
```

Arguments

x	character sting containing plantuml code.

Value

object of class `plantuml` which can be plotted.

Examples

```
## Not run:
x <- '
(*) --> "Initialization"

if "Some Test" then
-->[true] "Some Activity"
--> "Another activity"
-right-> (*)
else
  ->[false] "Something else"
-->[Ending process] (*)
endif
'
x <- plantuml( x )
plot( x )

## End(Not run)
```

`plantuml_file` *Convert file to plantuml code*

Description

The function either plots the object including values (`yaml` and `json`) or loads the object and plots the structure (`csv` and `rds`).

Exactly one of the two arguments (`file` or `text`) needs tobe specified.

Exactly one of the two arguments (`file` or `text`) needs tobe specified.

This is a convenience function which only reads the `csv` file using `read.csv(file, ...)` and converts the resulting `data.frame` to a plantuml object using `plantuml()`.

This is a convenience function which only reads the `rds` file using `readRDS(file, ...)` and converts the resulting object to a plantuml object using `plantuml()`.

Usage

```
plantuml_file(file, preamble = "", ...)

plantuml_yaml(file, text, preamble = "", ...)

plantuml_json(file, text, preamble = "", ...)

plantuml_csv(file, ...)
```

```
plantuml_rds(file, ...)
```

Arguments

file	file name of the rds file. The function does **not** do any checking if the file is a rds file!
preamble	text to be inserted after the @startyaml and before the actual yaml. For example style and highlight info. See https://plantuml.com/yaml for further info.
...	additional arguments. Will be passed to readRDS()
text	yaml text to be converted. The function does **not** do any checking if the file is valid yaml!

Value

a plantuml containing the yaml file for plotting

a plantuml containing the json file for plotting

a plantuml containing the structure of the csv file for plotting

a plantuml containing the structure of the rds file for plotting

Examples

```
## Not run:
plantuml_file("name.yml")
plantuml_file("name.yaml")
plantuml_file("name.json")
plantuml_file("name.rds")
plantuml_file("name.csv")

## End(Not run)
## some preparations
x1 <- "name: Test yml\na:\n- a\n- d\n- c\nB:\n- C\n- D\n- E\nx:\n  one: 0.2885\n  two: 0.7498\n"
x2 <- "name: Test yml\na:\n- a\n- b\n- c\nB:\n- C\n- D\n- E\nx:\n  one: 0.2865\n  two: 0.7498\n"
fn1 <- tempfile(fileext = ".yml")
fn2 <- tempfile(fileext = ".yml")
writeLines(x1, fn1)
writeLines(x2, fn2)

## and now the example

plot(plantuml_yaml(fn1))

## Now let's see the differences between `fn1` and `fn2`
## this requires the package `yaml` to be installed

if (require(yaml)) {
  plot(
    plantuml_yaml(
      file = fn1,
      preamble = diff_yaml_json(yaml::read_yaml(fn1), yaml::read_yaml(fn2))
```

```
   )
  )
}

## and cleanup

unlink(fn1)
unlink(fn2)

## some preparations
x <- '{"name":["Test json"],"a":["a","b","c"],"B":["C","D","E"],"x":[0.6464,0.6879]}'
fn <- tempfile(fileext = ".json")
writeLines(x, fn)

## and now the example

plot(plantuml_json(fn))

## and cleanupo

unlink(fn)

## some preparations
x <- data.frame(
  a = c("a", "b", "c"),
  B = c("C", "D", "E"),
  x = c(0.77, 0.38, 4.43),
  bool = c(TRUE, FALSE, FALSE)
)

fn <- tempfile(fileext = ".csv")
write.csv(x, fn)

## and now the example

plot(plantuml_csv(fn))

# and only the fors columns (the rownames will not all be there)

plot(plantuml_csv(fn, nrows = 1))

# or character vectors as factors

plot(plantuml_csv(fn, as.is = FALSE))

## and cleanupo

unlink(fn)

## some preparations
x <- list(
  name = "Test list",
  a = c("a", "b", "c"),
```

```
  B = c("C", "D", "E"),
  x = c(0.776318477466702, 0.381654617609456)
)

fn <- tempfile(fileext = ".rds")
saveRDS(x, fn)

## and now the example

plot(plantuml_rds(fn))

## and cleanupo

unlink(fn)
```

`plantuml_knit_engine` *Knit engine for plantuml*

Description

This function provides a knit engine for the plantuml. It has the following additional functions:

- **plantuml.format**: the format of the generated image. For formats which return images, these will be inserted, formats resulting in text qill be included as text. At the moment, the following are supported:
  - **auto** (default) uses `svg` if output is `html` and `pdf` if output is `pdf`
  - **png** To generate image using PNG format (default).
  - **svg** To generate image using SVG format.
  - **eps** To generate text in EPS format; or generates an image when outputting LaTeX rather than HTML formats.
- **plantuml.path**: the path where the resulting files will be saved. Default is the same directory as the `.Rmd` file is in in a directory named ‘ plantuml. The path will be created if it does not exist. The name of each plantuml figure is the label of thre chunk with the extension as specified.
- **plantuml.preview**: if `TRUE`, an inline preview will be shown in RStudio. **Attention: the processing takse twice as long as without this option!**.

Usage

```
plantuml_knit_engine(options)
```

Arguments

`options`	knitr options see `knitr`

Details

Thanks to Emiliano Heyns (retorquere) for giving me the idea for this functiion https://github.com/rkrug/plantuml/issues/10#i
639795529 See `knit_engines` in the knitr packages for details

Value

See `knit_engines` in the knitr packages for details

Examples

```
## Not run:
## see
system.file("plantuml.Rmd", package = "plantuml")
## for an example RMarkdown file usint this knit engine.

## End(Not run)
```

---

plantuml_knit_engine_register

Register the function `plantuml_knit_engine()` *with knitr*

Description

This is a simple helper function, which registers the plantuml knit engine with knitr. As a result, code chunks of type `plantuml` can be evaluated.

Usage

```
plantuml_knit_engine_register()
```

Value

the result of `knitr::knit_engines$set(plantuml = plantuml::plantuml_knit_engine)`

Examples

```
## Not run:
# in the setup chunk in a RMarkdown document, add
plantuml_knit_engine_register()

## End(Not run)
```

plantuml_run *Run the plantuml binary*

Description

The in the package installation included plantuml binary is executed using the provided java and plantuml commands. This is effectively a wrapper around system2() with some values set to run plantuml.

Usage

```
plantuml_run(
  x = NULL,
  file = "",
  plantuml_jar = getPlantumlOption("jar_name"),
  plantuml_opt = getPlantumlOption("plantuml_opt"),
  java_bin = getPlantumlOption("java_bin"),
  java_opt = getPlantumlOption("java_opt"),
  wait = FALSE
)
```

Arguments

x	plantuml code to draw the UML graph
file	file name, including extension, to which the generated plantUML graph should be saved. The extension determines the format of the graph. If NULL', the graph is returned as a ASC acter‘ vector..
plantuml_jar	path and name of the plantuml jar file. The dafault is read from getPlantumlOption("plantuml_jar").
plantuml_opt	options for the call of java. The default is read from getPlantumlOption("plantuml_opt").
java_bin	path to the java binary. The dafault is read from getPlantumlOption("java_bin").
java_opt	options for the call of java. The default is read from getPlantumlOption("java_opt").
wait	if TRUE, wait until the process has finished. If FALSE, return immediately.

Value

if wait is TRUE the pid of the process started. Otherwise the result from the call to system2()

Examples

```
## Not run:
# This will take some time when you run it
# for the first time as it will download \code{plantuml.jar}
  plantuml_run()

## End(Not run)
```

plantuml_update *Update or download* plantuml.jar *binary*

Description

The file plantuml.jar is downloaded from Sourceforge and saved as "plantuml.jar" in the folder system.file("jar", package = "plantuml") of the package. The source code for plantuml can be found at https://github.com/plantuml/plantuml

Usage

```
plantuml_update(tag = "release", ...)
```

Arguments

tag	tag of the version to be downloaded. Allowed values are • **"release** : the last release • **"snapshot"** : the last snapshot - **Not ready for gneral use**
...	additional arguments for the download.file() function

Value

the path and name of the downloaded file

Examples

```
## Not run:
plantuml_update()

## End(Not run)
```

plantuml_URL *Generate PlantUML Server URL*

Description

Generate PlantUML Server URL

Usage

```
plantuml_URL(
  plantuml = "@startuml\nBob -> Alice : hello\n@enduml",
  server_url = getPlantumlOption("server_url"),
  server_port = getPlantumlOption("server_port"),
  type = "svg",
  open_in_browser = FALSE
)
```

Arguments

plantuml	The plantuml code. If NULL, only the base URL consisting of server and (if specified) port will be returned.
server_url	The base url of the server. Should end with a single / If NULL, encoding, including the ttpe, will be returned.
server_port	port on which the plantuml server is
type	the type of the returned figure. At the moment supported: png, svg txt, or map. See https://plantuml.com/server for further details. If NULL, encoding, excluding the tytpe, will be returned. **The return value is invalid if** server_url **is given!**
open_in_browser	if TRUE, the result will be opened in the browser.

Value

complete url to retrieve the graph.

---

plot.plantuml *Generate UML graph from plantuml*

Description

Generate an image containing based ion the plantuml code. TODO can I use vector formats?

Usage

```
## S3 method for class 'plantuml'
plot(x, file = NULL, width = 1024, height = NULL, css = NULL, ...)
```

Arguments

x	plantuml code to draw the UML graph
file	• file **is 'NULL**: the graph is drawn in the graphics device. • file **is a file name**: the graph is saved in the file and the type is based on the extensions. • file **is** NULL: the data which would have been saved in the file is returned in a character vector. **This is only useful for text formats like** eps **or** svg**!**
width	output width in pixels or NULL for default.
height	output height in pixels or NULL for default
css	path/url to external css file or raw vector with css data. This requires your system has a recent version of librsvg.
...	additional arguments for the plot function and the plantuml_run function.

Value

returns file name (including absolute path) of the created graph.

Examples

```
plantumlCode <- '
 @startuml
 (*) --> "First Activity"
 -->[You can put also labels] "Second Activity"
 --> (*)
 @enduml
'
## Not run:
x <- as.plantuml( plantumlCode )
plot( x )
plot(as.plantuml(x), java_opt = "-Djava.awt.headless=true")

## End(Not run)
```

---

server_info *Get info about the plantuml server*

Description

The parameter (URL and port) as in the `getPlantumlOption()` specified are used.

Usage

```
server_info()
```

Value

list with the following arguments:

- **server**: type of the server, **PicoWeb Server** or **PlantUML Web Server**
- **supported_formats: supported formats of the server

server_set *Set parameter of the plantuml server addrss to the default valus*

Description

Set parameter of the plantuml server addrss to the default valus

Usage

```
server_set(location = "remote")
```

Arguments

location	alloeed values are: • "remote": default values for the remote server • "local": default values for a local picoweb server. The serer still needs to be started! Other locations need to be set manually by setting the options directly.

Value

invisibly the old values of plantuml options

server_start *Start a PlantUML Picoweb Server on localhost*

Description

If the port `getPlantumlOption("server_port")` on localhostis is available, i.e. no other server is running on that port, a PantUML PicoWeb Server is started. The return value indicates if a server was started and if not, why.

Usage

```
server_start()
```

Value

`integer` or `list()` as returned from `server_status()`

- **-999**: unknown error
- **1** : unknown server running on the port on localhost
- **2** : PlantUML Web Server running on that port on localhost
- `server_info()` ehen PicoWeb Server already running or started

server_status *Get the status of the server*

Description

The parameter (URL and port) as in the `getPlantumlOption()` specified are used.

Usage

```
server_status()
```

Value

`integer` value indicating the status:

- **-999**: undetermined error (should not happen)!
- **-404**: URL not reachable
- **0** : server reachable but not a known PlantUml Web Server or PicWeb Plantuml Server
- **1** : server is PicoWeb Server
- **2** : server is PlantUML Web Server

server_stop *Stop a PlantUML Picoweb Server running on localhost*

Description

The server is expscted at the port `getPlantumlOption("server_port")` on localhost.

Usage

```
server_stop()
```

Value

`integer` or `list()` as returned from `server_status()`

- **-999**: unknown error
- **-1** : unknown server running on the port on localhost
- **-2** : PlantUML Web Server running on that port on localhost
- **0** : PicoWeb Server in the process of stopping or no server running on tat port

`updatePlantumlJar` *RENAMED to plantuml_update()!*

Description

RENAMED to plantuml_update()!

Usage

```
updatePlantumlJar(...)
```

Arguments

...	some parameter

Index