Package ‘pander’

February 20, 2015

Maintainer  Gergely Daróczi <daroczig@rapporter.net>
Title      An R Pandoc Writer
Type      Package
Encoding  UTF-8
Description Contains some functions catching all messages, stdout and other useful information while evaluating R code and other helpers to return user specified text elements (like: header, paragraph, table, image, lists etc.) in Pandoc’s markdown or several type of R objects similarly automatically transformed to markdown format. Also capable of exporting/converting (the resulting) complex Pandoc documents to e.g. HTML, PDF, docx or odt. This latter reporting feature is supported in brew syntax or with a custom reference class with a smarty caching backend.

Author  Gergely Daróczi <daroczig@rapporter.net>, Roman Tsegelskyi <roman.tsegelskyi@gmail.com>

Version  0.5.1
Date     2014-10-29

URL     http://rapporter.github.io/pander

BugReports https://github.com/rapporter/pander/issues

License  AGPL-3 | file LICENSE

Depends  R (>= 2.15.0)
Imports  methods, digest, tools, Rcpp
Suggests  grid, lattice, ggplot2 (>= 0.9.2), koRpus, survival, microbenchmark, zoo, nlme, descr, MASS, knitr

SystemRequirements Pandoc (http://johmacfarlane.net/pandoc) for exporting markdown files to other formats.

LinkingTo  Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-29 10:02:39
R topics documented:

add.blank.lines ............................................. 4
add.significance.stars .................................... 4
cache.off ..................................................... 5
emphasize.rows .............................................. 5
eval.msgs ..................................................... 6
evals .......................................................... 8
evalsOptions .................................................. 15
has.rownames ................................................ 17
openFileInOS .................................................. 18
p ................................................................. 18
pander .......................................................... 19
pander.anova .................................................. 21
pander.aov .................................................... 22
pander.aovlist ............................................... 22
pander.call .................................................... 23
pander.cast_df ............................................... 23
pander.character ............................................ 24
pander.clogit ............................................... 24
pander.coxph ................................................ 25
pander.CrossTable .......................................... 25
pander.data.frame .......................................... 26
pander.Date .................................................. 26
pander.default .............................................. 27
pander.density .............................................. 27
pander.describe ............................................ 28
pander.evals ............................................... 28
pander.factor ............................................... 29
pander.formula ............................................. 29
pander.ftable ............................................... 30
pander.function ............................................ 30
pander.glm ................................................... 31
pander.htest ............................................... 31
pander.image ............................................... 32
pander.list .................................................. 32
pander.lm .................................................... 33
pander.lme ................................................... 33
pander.logical ............................................... 34
pander.matrix .............................................. 34
pander.microbenchmark ................................... 35
pander.mtable .............................................. 35
pander.NULL ................................................ 36
pander.numeric ............................................. 36
pander.POSIXct ............................................. 37
pander.POSIXlt ............................................ 37
pander.prcomp ............................................. 38
pander.rapport ............................................. 38
R topics documented:

pander.rlm ................................................................. 39
pander.sessionInfo .................................................... 39
pander.smooth.spline .................................................. 40
pander.stat.table ........................................................ 40
pander.summary.aov ..................................................... 41
pander.summary.aovlist ................................................ 41
pander.summary.glm ..................................................... 42
pander.summary.lm ....................................................... 42
pander.summary.prcomp ................................................ 43
pander.survdiff .......................................................... 44
pander.survfit ........................................................... 44
pander.table .............................................................. 45
pander.ts ................................................................. 45
pander.zoo ............................................................... 46
panderOptions ........................................................... 46
Pandoc-class ............................................................. 49
Pandoc.brew .............................................................. 50
Pandoc.convert .......................................................... 52
pandoc.date.return ..................................................... 54
pandoc.emphasis.return ............................................... 54
pandoc.footnote.return ................................................. 55
pandoc.formula.return .................................................. 56
pandoc.header.return ................................................... 56
pandoc.horizontal.rule.return ....................................... 57
pandoc.image.return .................................................... 58
pandoc.indent ............................................................. 59
pandoc.link.return ...................................................... 59
pandoc.list.return ...................................................... 60
pandoc.p.return ......................................................... 61
pandoc.strikeout.return ............................................... 62
pandoc.strong.return ................................................... 62
pandoc.table.return .................................................... 63
pandoc.title.return ..................................................... 67
pandoc.verbatim.return ................................................. 68
redraw.recordedplot .................................................... 69
redrawPlot ............................................................... 70
remove.extra.newlines .................................................. 70
repChar ................................................................. 71
set.alignment ............................................................. 71
set.caption ............................................................... 72
splitLine ................................................................. 72
trim.spaces .............................................................. 73
wrap ................................................................. 73

Index 75
### add.blank.lines

*Add trailing and leading blank line*

**Description**

Adds a line break before and after the character string(s).

**Usage**

```r
add.blank.lines(x)
```

**Arguments**

- `x`: character vector

### add.significance.stars

*Add significance stars*

**Description**

This function adds significance stars to passed p value(s) as: one star for value below 0.05, two for 0.01 and three for 0.001.

**Usage**

```r
add.significance.stars(p)
```

**Arguments**

- `p`: numeric vector or tabular data

**Value**

- character vector
Description

This function is just a wrapper around `evalsOptions` to switch pander’s cache on or off easily, which might be handy in some brew documents to prevent repetitive strain injury :)

Usage

- `cache.on()`
- `cache.off()`

emphasize.rows  Emphasize rows/columns/cells

Description

Storing indexes of cells to be (strong) emphasized of a tabular data in an internal buffer that can be released and applied by `pandoc.table`, `pander` or `evals` later.

Usage

- `emphasize.rows(x)`
- `emphasize.cols(x)`
- `emphasize.cells(x)`
- `emphasize.strong.rows(x)`
- `emphasize.strong.cols(x)`
- `emphasize.strong.cells(x)`

Arguments

- `x`  vector of row/columns indexes or an array like returned by `which(..., arr.ind = TRUE)`
eval.msgs  Evaluate with messages

Description

This function takes text(s) of R code and evals all at one run - returning a list with four elements. See Details.

Usage

eval.msgs(src, env = NULL, showInvisible = FALSE, graph.unify = evalsOptions("graph.unify"))

Arguments

src  character values containing R code
env  environment where evaluation takes place. If not set (by default), a new temporary environment is created.
showInvisible  return invisible results?
graph.unify  should eval.msgs try to unify the style of (lattice and ggplot2) plots? If set to TRUE (by default), some panderOptions() would apply. Please note that this argument has no effect on base plots, use evals instead.

Details

eval.msgs returns a detailed list of the result of evaluation:

- src - character vector of specified R code.
- result - result of evaluation. NULL if nothing is returned. If any R code returned an R object while evaluating then the last R object will be returned as a raw R object. If a graph is plotted in the end of the given R code (remember: last R object), it would be automatically printed (see e.g. lattice and ggplot2).
- output - character vector of printed version (capture.output) of result
- type - class of generated output. "NULL" if nothing is returned, "error" if some error occurred.

Examples

## Not run:

n <- data.frame(x = c(1,1,1,1), y = c(0,1,0,1,0))
emphasize.cols(1)
emphasize.rows(1)
pandoc.table(n)

emphasize.strong.cells(which(n == 1, arr.ind = TRUE))
pander(n)

## End(Not run)
• **msg** - possible messages grabbed while evaluating specified R code with the following structure:
  – **messages** - character vector of possible diagnostic message(s)
  – **warnings** - character vector of possible warning message(s)
  – **errors** - character vector of possible error message(s)

• **stdout** - character vector of possibly printed texts to standard output (console)

**Value**

a list of parsed elements each containing: **src** (the command run), **result** (R object: NULL if nothing returned), **printed output**, **type** (class of returned object if any), informative/warning and error messages (if any returned by the command run, otherwise set to NULL) and possible stdout value. See Details above.

**See Also**

evals

**Examples**

```r
## Not run:
eval.msgs('1:5')
eval.msgs('x <- 1:5')
eval.msgs('lm(mtcars$hp - mtcars$wt)')

## plots
eval.msgs('plot(runif(100))')
eval.msgs('histogram(runif(100))')

## error handling
eval.msgs('runif(23)')
eval.msgs('runif is a nice function')
eval.msgs('no.R.object.like.that')

## messages
eval.msgs(c('message("FOO")', '1:2'))
eval.msgs(c('warning("FOO")', '1:2'))
eval.msgs(c('message("FOO")';message("FOO");warning("FOO"), '1:2'))
eval.msgs('warning("d");warning("f");1')

## stdout
eval.msgs('cat("writing to console")')
eval.msgs('cat("writing to console");1:4')
```

## End(Not run)
Description

This function takes either a vector/list of strings with actual R code, which it to be parsed to separate elements. Each list element is evaluated in a special environment, and a detailed list of results is returned for each logical part of the R code: a character value with R code, resulting R object, printed output, class of resulting R object, possible informative/warning/error messages and anything written to stdout. If a graph is plotted in the given text, the returned object is a string specifying the path to the saved file. Please see Details below. If parse option set to FALSE, then the returned list’s length equals to the length of the parsed input - as each string is evaluated as separate R code in the same environment. If a nested list of R code or a concatenated string (separated by \n or \) is provided like list(c('runif(1)', 'runif(1)')) with parse=FALSE, then everything is evaluated at one run so the length of returned list equals to one or the length of the provided nested list. See examples below.

Usage

evals(txt, parse = TRUE, cache = TRUE, cache.mode = c("environment", "disk"), cache.dir = ".cache", cache.time = 0.1, cache.copy.images = FALSE, showInvisible = FALSE, classes = NULL, hooks = NULL, length = Inf, output = c("all", "src", "result", "output", "type", "msg", "stdout"), env = NULL, graph.unify = evalsOptions("graph.unify"), graph.name = "%t", graph.dir = "plots", graph.output = c("png", "bmp", "jpeg", "jpg", "tiff", "svg", "pdf", NA), width = 480, height = 480, res = 72, hi.res = FALSE, hi.res.width = 960, hi.res.height = 960 * (height/width), hi.res.res = res * (hi.res.width/width), graph.env = FALSE, graph.recordplot = FALSE, graph.RDS = FALSE, ...)

Arguments

txt a character vector containing R code. This could be a list/vector of lines of code or a simple string holding R code separated by ; or \n.

parse if TRUE the provided txt elements would be merged into one string and parsed to logical chunks. This is useful if you would want to get separate results of your code parts - not just the last returned value, but you are passing the whole script in one string. To manually lock lines to each other (e.g. calling a plot and on next line adding an abline or text to it), use a plus char (+) at the beginning of each line which should be evaluated with the previous one(s). If set to FALSE, evals would not try to parse R code, it would get evaluated in separate runs - as provided. Please see examples below.

cache caching the result of R calls if set to TRUE. Please note the caching would not work if parse set to FALSE or syntax error is to be found.

cache.mode cached results could be stored in an environment in current R session or let it be permanent on disk.
cache.dir path to a directory holding cache files if cache.mode set to disk. Default to
.cache in current working directory.
cache.time number of seconds to limit caching based on proc.time. If set to 0, all R
commands, if set to Inf, none is cached (despite the cache parameter).
cache.copy.images copy images to new file names if an image is returned from the disk cache? If
set to FALSE (default), the cached path would be returned.
showInvisible return invisible results?
classes a vector or list of classes which should be returned. If set to NULL (by default)
all R objects will be returned.
hooks list of hooks to be run for given classes in the form of list(class = fn). If
you would also specify some parameters of the function, a list should be pro-
vided in the form of list(fn, param1, param2=NULL) etc. So the hooks
would become list(class=list(fn, param1, param2=NULL), ...). See
example below. A default hook can be specified too by setting the class to
'default'. This can be handy if you do not want to define separate meth-
ods/functions to each possible class, but automatically apply the default hook
to all classes not mentioned in the list. You may also specify only one element
in the list like: hooks=list('default' = pandeR.return). Please note, that
nor error/warning messages, nor stdout is captured (so: updated) while running
hooks!
length any R object exceeding the specified length will not be returned. The default
value (Inf) does not filter out any R objects.
output a character vector of required returned values. This might be useful if you are
only interested in the result, and do not want to save/see e.g. messages or
printed output. See examples below.
env environment where evaluation takes place. If not set (by default), a new tempo-
rary environment is created.
graph.unify should evals try to unify the style of (base, lattice and ggplot2) plots? If set
to TRUE, some pandeR.options() would apply. By default this is disabled not to
freak out useRs :)
graph.name set the file name of saved plots which is tempfile by default. A simple char-
acter string might be provided where %d would be replaced by the index of the
generating .txt source, %n with an incremented integer in graph.dir with sim-
ilar file names and %t by some unique random characters. While running in
Pandoc.brew other indices could be triggered like %i and %I.
graph.dir path to a directory where to place generated images. If the directory does not
exist, evals try to create that. Default set to plots in current working directory.
graph.output set the required file format of saved plots. Currently it could be any of grDevices':
png, bmp, jpeg, jpg, tiff, svg or pdf.
width width of generated plot in pixels for even vector formats
height height of generated plot in pixels for even vector formats
res nominal resolution in ppi. The height and width of vector images will be calcu-
lated based in this.
The generated image file(s) of the plots can be fine-tuned by some specific options, please check out `graph.output.width`, `graph.output.height`, `graph.output.res`, `graph.output.hi.res`, `graph.output.hi.res.width`, `graph.output.hi.res.height` and `graph.output.hi.res.res` parameters. Most of these options are better not to touch, see details of parameters below.

Returned result values: list with the following elements

- *src* - character vector of specified R code.
- *result* - result of evaluation. NULL if nothing is returned. If any R code returned an R object while evaluating then the last R object will be returned as a raw R object. If a graph is plotted in the given text, the returned object is a string (with `class` set to `image`) specifying the path to the saved image file. If graphic device was touched, then no other R objects will be returned.
- *output* - character vector of printed version (capture.output) of result
- *type* - class of generated output. "NULL" if nothing is returned, "error" if some error occurred.
- *msg* - possible messages grabbed while evaluating specified R code with the following structure:
  - *messages* - character vector of possible diagnostic message(s)
  - *warnings* - character vector of possible warning message(s)
  - *errors* - character vector of possible error message(s)
- *stdout* - character vector of possibly printed texts to standard output (console)
By default `evals` tries to cache results. This means that if evaluation of some R commands take too much time (specified in `cache.time` parameter), then `evals` would save the results in a file and return from there on next exact R code's evaluation. This caching algorithm tries to be smart as checks not only the passed R sources, but all variables inside that and saves the hash of those.

Technical details of the caching algorithm:

- Each passed R chunk is parsed to single commands.
- Each parsed command's part (let it be a function, variable, constant etc.) evaluated (as a name) separately to a list. This list describes the unique structure and the content of the passed R commands, and has some IMHO really great benefits (see examples below).
- A hash if computed to each list element and cached too in `pander`'s local environments. This is useful if you are using large data frames, just imagine: the caching algorithm would have to compute the hash for the same data frame each time it's touched! This way the hash is recomputed only if the R object with the given name is changed.
- The list is serialized and an SHA-1 hash is computed for that - which is unique and there is no real risk of collision.
- If `evals` can find the cached results in a file named to the computed hash, then it is returned on the spot.
- Otherwise the call is evaluated and the results are optionally saved to cache (e.g. if cache is active, if the `proc.time()` of the evaluation is higher then it is defined in `cache.time` etc.).

This is a quite secure way of caching, but if you would encounter any issues, just set cache to `FALSE` or tweak other cache parameters. While setting `cache.dir`, please do think about what you are doing and move your `graph.dir` accordingly, as `evals` might result in returning an image file path which is not found any more on your file system!

Also, if you have generated a plot and rendered that to e.g. `png` before and later try to get e.g. `pdf` - it would fail with cache on. Similarly you cannot render a high resolution image of a cached image, but you have to (temporary) disable caching.

The default `evals` options could be set globally with `evalsOptions`, e.g. to switch off the cache just run `evalsOptions(c('cache', FALSE))`.

Please check the examples carefully below to get a detailed overview of `evals`.

### Value

A list of parsed elements each containing: `src` (the command run), `result` (R object: `NULL` if nothing returned, path to image file if a plot was generated), `printed` output, `type` (class of returned object if any), informative/warning and error messages (if any returned by the command run, otherwise set to `NULL`) and possible `stdout` value. See Details above.

### See Also

`eval.msgs` `evalsOptions`
Examples

```r
## Not run:
# parsing several lines of R code
txt <- readLines(textConnection('
x <- rnorm(100)
runif(10)
warning("Lorem ipsum foo-bar-foo!")
plot(1:10)
qplot(rating, data = movies, geom = "histogram")
y <- round(runif(100))
cor.test(x, y)
crl <- cor.test(runif(10), runif(10))
table(mtcars$am, mtcars$cyl)
ggplot(mtcars) + geom_point(aes(x = hp, y = mpg)))
evals(txt)

## parsing a list of commands
txt <- list('df <- mtcars',
  c('plot(mtcars$hp, pch = 19)','text(mtcars$hp, label = rownames(mtcars), pos = 4)'),
  'ggplot(mtcars) + geom_point(aes(x = hp, y = mpg))')
evals(txt)

## the same commands in one string but also evaluating the `plot` with `text`
## (note the leading "+" on the beginning of `text...` line)
txt <- 'df <- mtcars
plot(mtcars$hp, pch = 19)
+text(mtcars$hp, label = rownames(mtcars), pos = 4)
ggplot(mtcars) + geom_point(aes(x = hp, y = mpg))'
evals(txt)

## but it would fail without parsing
evals(txt, parse = FALSE)

## handling messages
evals('message(20)')
evals('message(20);message(20)', parse = FALSE)

## adding a caption to a plot
evals('set.caption("FOO"); plot(1:10)')
## `plot` is started with a `+` to eval the codes in the same chunk
## (no extra chunk with NULL result)
evals('set.caption("FOO"); +plot(1:10)')

## handling warnings
evals('chisq.test(mtcars$gear, mtcars$hp)')
evals(list(c('chisq.test(mtcars$gear, mtcars$am)', 'pi',
  'chisq.test(mtcars$gear, mtcars$hp)')))
evals(c('chisq.test(mtcars$gear, mtcars$am)',
  'pi',
  'chisq.test(mtcars$gear, mtcars$hp)'))

## handling errors
evals('runif(20)')
evals('Old MacDonald had a farm\...')
```
evals('## Some comment')
evals(c('runif(20)', 'Old MacDonald had a farm?'))
evals(list(c('runif(20)', 'Old MacDonald had a farm?'), parse = FALSE)
evals(c('mean(1:10)', 'no.R.function()'))
evals(list(c('mean(1:10)', 'no.R.function()'), parse = FALSE)
evals(c('no.R.object', 'no.R.function()', 'very.mixed.up(stuff)'))
evals(list(c('no.R.object', 'no.R.function()', 'very.mixed.up(stuff)'), parse = FALSE)
evals(c('no.R.object', 'Old MacDonald had a farm\ldots', 'pi'))
evals(list(c('no.R.object', 'Old MacDonald had a farm\ldots', 'pi'), parse = FALSE)

## graph options
evals('plot(1:10)')
evals('plot(1:10);plot(2:20)')
evals('plot(1:10)', graph.output = 'jpg')
evals('plot(1:10)', height = 800)
evals('plot(1:10)', height = 800, hi.res = TRUE)
evals('plot(1:10)', graph.output = 'pdf', hi.res = TRUE)
evals('plot(1:10)', res = 30)
evals('plot(1:10)', graph.name = 'myplot')
evals(list('plot(1:10)', 'plot(2:20)'), graph.name = 'myplots-kd')
evals('plot(1:10)', graph.env = TRUE)
evals('x <- runif(100);plot(x)', graph.env = TRUE)
evals(c('plot(1:10)', 'plot(2:20)'), graph.env = TRUE)
evals(c('x <- runif(100)', 'plot(x)', 'y <- runif(100)', 'plot(y)'), graph.env = TRUE)
evals(list(  c('x <- runif(100)', 'plot(x)'),
            c('y <- runif(100)', 'plot(y)'),
            graph.env = TRUE, parse = FALSE)
evals('plot(1:10)', graph.recordplot = TRUE)

## unprinted lattice plot
evals('histogram(mtcars$hp)', graph.recordplot = TRUE)

## caching
system.time(evals('plot(mtcars)'))         # running again to see the speed-up :)
system.time(evals('plot(mtcars)'), cache = FALSE)) # cache disabled

## caching mechanism does check what's inside a variable:
x <- mtcars
evals('plot(x)')
x <- cbind(mtcars, mtcars)
evals('plot(x)')
x <- mtcars
system.time(evals('plot(x)'))

## stress your CPU - only once!
evals('x <- sapply(rep(mtcars$hp, 1e3), mean)') # run it again!

## play with cache
require(lattice)
evals('histogram(rep(mtcars$hp, 1e5)')

## nor run the below call
## that would return the cached version of the above call:
```r
f <- histogram
g <- rep
A <- mtcars$hp
B <- 1e5
evals('f(g(A, B))')#
```

## or switch off cache globally:
evalsOptions('cache', FALSE)
## and switch on later
evalsOptions('cache', TRUE)

## returning only a few classes
txt <- readLines(textConnection('rnorm(100)
  list(x = 10:1, y = "Godzilla!")
  c(1,2,3)
  matrix(0,3,5)'))
evals(txt, classes = 'numeric')
evals(txt, classes = c('numeric', 'list'))

## hooks
txt <- 'runif(1:4); matrix(runif(25), 5, 5); 1:5'
hooks <- list('numeric' = round, 'matrix' = pander.return)
evals(txt, hooks = hooks)
## using pander's default hook
evals(txt, hooks = list('default' = pander.return))
evals('22/7', hooks = list('numeric' = round))
evals('matrix(runif(25), 5, 5)', hooks = list('matrix' = round))

## setting default hook
evals(c('runif(10)', 'matrix(runif(9), 3, 3)'),
  hooks = list('default'=round))
## round all values except for matrices
evals(c('runif(10)', 'matrix(runif(9), 3, 3)'),
  hooks = list(matrix = 'print', 'default' = round))

# advanced hooks
hooks <- list('numeric' = list(round, 2), 'matrix' = list(round, 1))
evals(txt, hooks = hooks)

# return only returned values
evals(txt, output = 'result')

# return only messages (for checking syntax errors etc.)
evals(txt, output = 'msg')

# check the length of returned values and do not return loooong R objects
evals('runif(10)', length = 5)

# note the following will not be filtered!
evals('matrix(1,1,1)', length = 1)

# if you do not want to let such things be eval-ed in the middle of a string
# use it with other filters:
```r
evals('matrix(1,1,1)', length = 1, classes = 'numeric')
```

# hooks & filtering
```r
evals('matrix(5,5,5)',
    hooks = list('matrix' = pander.return),
    output = 'result')
```

# eval-ing chunks in given environment
```r
myenv <- new.env()
evals('x <- c(0,10)', env = myenv)
evals('mean(x)', env = myenv)
rm(myenv)
```

# note: if you had not specified 'myenv', the second 'evals' would have failed
```r
evals('x <- c(0,10)')
evals('mean(x)')
```

## Evalsoptions

### Querying/setting evals option

#### Description

To list all evals options, just run this function without any parameters provided. To query only one value, pass the first parameter. To set that, use the value parameter too.

#### Usage

`evalsOptions(o, value)`

#### Arguments

- `o` option name (string). See below.
- `value` value to assign (optional)

#### Details

The following evals options are available:

- **parse**: if TRUE the provided txt elements would be merged into one string and parsed to logical chunks. This is useful if you would want to get separate results of your code parts - not just the last returned value, but you are passing the whole script in one string. To manually lock lines to each other (e.g. calling a plot and on next line adding an abline or text to it), use a plus char (+) at the beginning of each line which should be evaluated with the previous one(s). If set to FALSE, evals would not try to parse R code, it would get evaluated in separate runs - as provided. Please see examples of `evals`.
- **cache**: caching the result of R calls if set to TRUE
• cache.mode: cached results could be stored in an environment in *current* R session or let it be permanent on disk.
• cache.dir: path to a directory holding cache files if cache.mode set to disk. Default to .cache in current working directory.
• cache.time: number of seconds to limit caching based on proc.time. If set to 0, all R commands, if set to Inf, none is cached (despite the cache parameter).
• cache.copy.images: copy images to new files if an image is returned from cache? If set to FALSE (default) the "old" path would be returned.
• classes: a vector or list of classes which should be returned. If set to NULL (by default) all R objects will be returned.
• hooks: list of hooks to be run for given classes in the form of list(class = fn). If you would also specify some parameters of the function, a list should be provided in the form of list(fn, param1, param2=NULL) etc. So the hooks would become list(class1=list(fn, param1, param2=NULL)). See examples of *evals*. A default hook can be specified too by setting the class to 'default'. This can be handy if you do not want to define separate methods/functions to each possible class, but automatically apply the default hook to all classes not mentioned in the list. You may also specify only one element in the list like: hooks=list('default' = pander.return).
Please note, that nor error/warning messages, nor stdout is captured (so: updated) while running hooks!
• length: any R object exceeding the specified length will not be returned. The default value (Inf) does not filter out any R objects.
• output: a character vector of required returned values. This might be useful if you are only interested in the result, and do not want to save/see e.g. messages or printed output. See examples of *evals*.
• graph.unify: should *evals* try to unify the style of (base, lattice and ggplot2) plots? If set to TRUE, some panderOptions() would apply. By default this is disabled not to freak out useRs :)
• graph.name: set the file name of saved plots which is *tempfile* by default. A simple character string might be provided where %d would be replaced by the index of the generating txt source, %n with an incremented integer in graph.dir with similar file names and %t by some random characters. A function's name to be evaluated can be passed here too.
• graph.dir: path to a directory where to place generated images. If the directory does not exist, *evals* try to create that. Default set to plots in current working directory.
• graph.output: set the required file format of saved plots. Currently it could be any of grDevices: png, bmp, jpeg, jpg, tiff, svg or pdf. Set to NA not to save plots at all and tweak that setting with capture.plot() on demand.
• width: width of generated plot in pixels for even vector formats
• height: height of generated plot in pixels for even vector formats
• res: nominal resolution in ppi. The height and width of vector images will be calculated based in this.
• hi.res: generate high resolution plots also? If set to TRUE, each R code parts resulting an image would be run twice.
• hi.res.width: width of generated high resolution plot in pixels for even vector formats. The height and res of high resolution image is automatically computed based on the above options to preserve original plot aspect ratio.
has.rownames

- `graph.env`: save the environments in which plots were generated to distinct files (based on `graph.name`) with `env` extension?
- `graph.recordplot`: save the plot via `recordPlot` to distinct files (based on `graph.name`) with `recodplot` extension?
- `graph.RDS`: save the raw R object returned (usually with `lattice` or `ggplot2`) while generating the plots to distinct files (based on `graph.name`) with RDS extension?

**Note**

`evals.option` is deprecated and is to be removed in future releases.

**See Also**

`evals` `panderOptions`

**Examples**

```r
evalsOptions()
evalsOptions('cache')
evalsOptions('cache', FALSE)
```

---

<table>
<thead>
<tr>
<th>has.rownames</th>
<th>Check if rownames are available</th>
</tr>
</thead>
</table>

**Description**

Dummy helper to check if the R object has real rownames or not.

**Usage**

`has.rownames(x)`

**Arguments**

- `x` a tabular-like R object

**Value**

TRUE OR FALSE
openFileInOS | Open file

**Description**

Tries to open a file with operating system’s default program.

**Usage**

`openFileInOS(f)`

**Arguments**

- `f` file (with full path)

**References**

This function is a fork of David Hajage’s `convert` function: [https://github.com/eusebe/ascii/blob/master/R/export.r](https://github.com/eusebe/ascii/blob/master/R/export.r)

---

**p | Inline Printing**

**Description**

`p` merges elements of a vector in one string for the sake of pretty inline printing. Default parameters are read from appropriate option values (see argument description for details). This function allows you to put the results of an expression that yields a variable `inline`, by wrapping the vector elements with the string provided in `wrap`, and separating elements by main and ending separator (`sep` and `copula`). In case of a two-length vector, value specified in `copula` will be used as a separator. You can also control the length of provided vector by altering an integer value specified in `limit` argument (defaults to `Inf`).

**Usage**

```r
p(x, wrap = panderOptions("p.wrap"), sep = panderOptions("p.sep"),
    copula = panderOptions("p.copula"), limit = Inf,
    keep.trailing.zeros = panderOptions("keep.trailing.zeros"))
```
Arguments

- **x**: an atomic vector to get merged for inline printing
- **wrap**: a string to wrap vector elements (uses value set in `p.wrap` option; "_" by default, which is a markdown-friendly wrapper and it puts the string in *italic*)
- **sep**: a string with the main separator, i.e. the one that separates all vector elements but the last two (uses the value set in `p.sep` option - "," by default)
- **copula**: a string with ending separator - the one that separates the last two vector elements (uses the value set in `p.copula` option, "and" by default)
- **limit**: maximum character length (defaults to Infinitive elements)
- **keep.trailing.zeros**: to show or remove trailing zeros in numbers

Value

a string with concatenated vector contents

Author(s)

Aleksandar Blagotic

References

This function was moved from `rapport` package: http://rapport-package.info/.

Examples

```r
p(c("fee", "fi", "foo", "fam"))
#> [1] "fee", "fi", "foo" and "fam"
p(1:3, wrap = "")
#> [1] "1, 2 and 3"
p(LETTERS[1:5], copula = "and the letter")
#> [1] "A, B, C, D and the letter E"
p(c("Thelma", "Louise"), wrap = ",", copula = "&")
#> [1] "Thelma & Louise"
```

---

**pander**

Generic *pander* method

**Description**

Prints an R object in Pandoc’s markdown.

**Usage**

```r
pander(x = NULL, ...)
```
Arguments

- `x` an R object
- ... optional parameters passed to special methods and/or raw pandoc.* functions

Value

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

Note

This function can be called by `pander` and `pandoc` too.

References

- David Hajage (2011): _ascii. Export R objects to several markup languages._ http://CRAN.R-project.org/package=ascii

Examples

```r
## Vectors
pander(1:10)
pander(letters)
pander(mtcars$am)
pander(factor(mtcars$am))

## Lists
pander(list(1, 2, 3, c(1, 2)))
pander(list(a = 1, b = 2, c = table(mtcars$am)))
pander(list(1, 2, 3, list(1, 2)))
pander(list(a = 1, 2, 3, list(1, 2)))
pander(list('FOO', letters[1:3], list(1:5), table(mtcars$gear), list('FOOBAR', list('a', 'b'))))
pander(list(a = 1, b = 2, c = table(mtcars$am), x = list(myname = 1, 2), 56))
pander(unclass(chisq.test(table(mtcars$am, mtcars$gear))))

## Arrays
pander(mtcars)
pander(table(mtcars$am))
pander(table(mtcars$am, mtcars$gear))

## Tests
pander(ks.test(runif(50), runif(50)))
pander(chisq.test(table(mtcars$am, mtcars$gear)))
pander(t.test(extra ~ group, data = sleep))

## Models
ml <- with(lm(mpg ~ hp + wt), data = mtcars)
pander(ml)
pander(anova(ml))
pander(aov(ml))

## Dobson (1990) Page 93: Randomized Controlled Trial (examples from: ?glm)
counts <- c(18, 17, 15, 20, 14, 16, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
m <- glm(counts ~ outcome + treatment, family = poisson())
pander(m)
pander(anova(m))
pander(aov(m))

## overwriting labels
pander(lm(Sepal.Width ~ Species, data = iris), covariate.labels = c('Versicolor', 'Virginica'))

## Prcomp
pander(prcomp(USArrests))

## Others
pander(density(runif(10)))
pander(density(mtcars$hp))

## default method
x <- chisq.test(table(mtcars$am, mtcars$gear))
class(x) <- 'I heave never heard of!'
pander(x)

---

**pander.anova**  
*Pander method for anova class*

**Description**

Prints an anova object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'anova'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: an anova object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to `raw_pandoc_table` function
pander.aov  

**Pander method for aov class**

**Description**

Prints an aov object in Pandoc's markdown.

**Usage**

```r
## S3 method for class 'aov'
pander(x, caption = attr(x, "caption"), ...)  
```

**Arguments**

- `x`: an aov object  
- `caption`: caption (string) to be shown under the table  
- `...`: optional parameters passed to raw pandoc.table function

---

pander.aovlist  

**Pander method for aovlist class**

**Description**

Prints an aovlist object in Pandoc's markdown.

**Usage**

```r
## S3 method for class 'aovlist'
pander(x, caption = attr(x, "caption"), ...)  
```

**Arguments**

- `x`: an aovlist object  
- `caption`: caption (string) to be shown under the table  
- `...`: optional parameters passed to raw pandoc.table function
**pander.call**  

**Pander method for call class**

**Description**

Prints a call object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'call'
pander(x, ...)
```

**Arguments**

- `x`: a call object
- `...`: optional parameters passed to raw `pandoc.formula` function

---

**pander.cast_df**  

**Pander method for cast_df class**

**Description**

Prints a cast_df object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'cast_df'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: a cast_df object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw `pandoc.table` function
pander.character  

*Pander method for character class*

**Description**

Prints a character class in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'character'
pander(x, ...)
```

**Arguments**

- `x`  
  a character object

- `...`  
  ignored parameters

---

pander.clogit  

*Pander method for clogit class*

**Description**

Prints a clogit object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'clogit'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`  
  an clogit object

- `caption`  
  caption (string) to be shown under the table

- `...`  
  optional parameters passed to raw `pandoc.table` function
**pander.coxph**  
*Pander method for coxph class*

**Description**

Prints a coxph object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'coxph'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`  
  an coxph object
- `caption`  
  caption (string) to be shown under the table
- `...`  
  optional parameters passed to raw pandoc.table function

---

**pander.CrossTable**  
*Pander method for CrossTable class*

**Description**

Prints a CrossTable object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'CrossTable'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`  
  a CrossTable object
- `caption`  
  caption (string) to be shown under the table
- `...`  
  optional parameters passed to raw pandoc.table function
### pander.data.frame

**Pander method for `data.frame` class**

**Description**

Prints a `data.frame` object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'data.frame'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x` a `data.frame` object
- `caption` caption (string) to be shown under the table
- `...` optional parameters passed to `raw_pandoc_table` function

### pander.Date

**Pander method for `Date` class**

**Description**

Prints a `Date` object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'Date'
pander(x, ...)
```

**Arguments**

- `x` a `Date` object
- `...` optional parameters passed to `raw_pandoc_date` function
pander.default  Default Pander method

Description
Method to be used, when no exact S3 method for given object is found. Tries to render object as a list.

Usage
## Default S3 method:
pander(x, ...)

Arguments
- x: an object
- ...: optional parameters passed to raw pandoc.list function

pander.density  Pander method for density class

Description
Prints a density object in Pandoc’s markdown.

Usage
## S3 method for class 'density'
pander(x, caption = attr(x, "caption"), ...)

Arguments
- x: a density object
- caption: caption (string) to be shown under the table
- ...: optional parameters passed to raw pandoc.table function
### pander.describe

**Pander method for describe class**

#### Description

Prints a describe object in Pandoc’s markdown.

#### Usage

```
## S3 method for class 'describe'
pander(x, caption = attr(x, "caption"), short = TRUE,
       split.tables = 60, ...)
```

#### Arguments

- **x**: an describe object
- **caption**: caption (string) to be shown under the table
- **short**: (default:TRUE) if to use concise output
- **split.tables**: (default:60) split.tables param for pandoc.table function
- **...**: optional parameters passed to raw pandoc.table function

### pander.evals

**Pander method for evals class**

#### Description

Prints a evals object in Pandoc’s markdown.

#### Usage

```
## S3 method for class 'evals'
pander(x, ...)
```

#### Arguments

- **x**: a evals object
- **...**: ignored parameters
### pander.factor

**Pander method for factor class**

**Description**

Prints a factor object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'factor'
pander(x, ...)
```

**Arguments**

- `x`: a factor object
- `...`: ignored parameters

### pander.formula

**Pander method for formula class**

**Description**

Prints a formula object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'formula'
pander(x, max.width = 80, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: a formula object
- `max.width`: maximum width in characters per line
- `caption`: caption (string) to be shown under the formula
- `...`: optional parameters passed to raw pandoc.table function
pander.ftable  

**Pander method for ftable class**

**Description**

Prints a ftable object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'ftable'
pander(x, ...)
```

**Arguments**

- `x`: a ftable object
- `...`: optional parameters passed to `raw_pandoc_table` function

pander.function  

**Pander method for function class**

**Description**

Prints an function object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'function'
pander(x, add.name = FALSE, verbatim = TRUE,
syntax_highlighting = FALSE, ...)
```

**Arguments**

- `x`: an function object
- `add.name` (default: `FALSE`) if to add function name to output or just to print a body
- `verbatim` (default: `TRUE`) if to add tabulation, so pandoc conversion will render it properly
- `syntax_highlighting` (default: `FALSE`) if to add highlighting tag for R syntax
- `...`: ignored parameters
### pander.glm

**Pander method for summary.glm class**

**Description**

Prints a summary.glm object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'glm'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x` : a summary.glm object
- `caption` : caption (string) to be shown under the table
- `...` : optional parameters passed to raw pandoc.table function

---

### pander.htest

**Pander method for htest class**

**Description**

Prints a htest object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'htest'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x` : a htest object
- `caption` : caption (string) to be shown under the table
- `...` : optional parameters passed to raw pandoc.table function
pander.image  
\textit{Pander method for image class}

\begin{description}
\item[Description] Prints a image object in Pandoc’s markdown.
\end{description}

\begin{description}
\item[Usage] 
\begin{verbatim}
## S3 method for class 'image'
pander(x, caption = attr(x, "caption"), href = attr(x, "href"), ...)
\end{verbatim}
\end{description}

\begin{description}
\item[Arguments]
\begin{itemize}
\item \textbf{x}  
a image object
\item \textbf{caption}  
caption (string) to be shown under the table
\item \textbf{href}  
link that image should be linked with
\item \textbf{...}  
ignored parameters
\end{itemize}
\end{description}

pander.list  
\textit{Pander method for list class}

\begin{description}
\item[Description] Prints a list object in Pandoc’s markdown.
\end{description}

\begin{description}
\item[Usage] 
\begin{verbatim}
## S3 method for class 'list'
pander(x, ...)
\end{verbatim}
\end{description}

\begin{description}
\item[Arguments]
\begin{itemize}
\item \textbf{x}  
a list object
\item \textbf{...}  
ignored parameters
\end{itemize}
\end{description}
pander.lm  

Pander method for summary.glm class

Description

Prints a summary.glm object in Pandoc’s markdown.

Usage

## S3 method for class 'lm'
pander(x, caption = attr(x, "caption"), covariate.labels, omit, ...)

Arguments

- `x` a summary.glm object
- `caption` caption (string) to be shown under the table
- `covariate.labels` vector to replace covariate lables in the table
- `omit` vector of variable to omit for priting in resulting table
- `...` optional parameters passed to raw pandoc.table function

pander.lme  

Pander method for lme class

Description

Prints a lme object in Pandoc’s markdown.

Usage

## S3 method for class 'lme'
pander(x, caption = attr(x, "caption"), summary = FALSE, ...)

Arguments

- `x` a lme object
- `caption` caption (string) to be shown under the table
- `summary` (default:FALSE) if to print expnder summary
- `...` optional parameters passed to raw pandoc.table function
pander.logical

Pander method for logical class

Description

Prints a logical object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'logical'
pander(x, ...)
```

Arguments

- `x` a logical object
- `...` ignored parameters

pander.matrix

Pander method for matrix class

Description

Prints a matrix object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'matrix'
pander(x, caption = attr(x, "caption"), ...)
```

Arguments

- `x` a matrix object
- `caption` caption (string) to be shown under the table
- `...` optional parameters passed to raw `pandoc.table` function
pander.microbenchmark  

Pander method for microbenchmark class

Description

Prints an microbenchmark object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'microbenchmark'
pander(x, caption = attr(x, "caption"), expr.labels, unit, ...)
```

Arguments

- `x`: an microbenchmark object
- `caption`: caption (string) to be shown under the table
- `expr.labels`: expression labels that will replace default ones (similar to rownames, which microbenchmark class table does not have)
- `unit`: units in which values should be printed (for example second, microseconds, etc.). Should be one of “ns”, “us”, “ms”, “s”, “t”, “hz”, “khz”, “mhz”, “eps”, “f”
- `...`: optional parameters passed to raw pandoc.table function

pander.mtable  

Pander method for mtable class

Description

Prints a mtable object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'mtable'
pander(x, caption = attr(x, "caption"), ...)
```

Arguments

- `x`: a mtable object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw pandoc.table function
## pander.NULL

### Description

Prints a NULL object in Pandoc’s markdown.

### Usage

```r
## S3 method for class 'NULL'
pander(x, ...)
```

### Arguments

- **x**: a NULL object
- **...**: ignored parameters

## pander.numeric

### Description

Prints a numeric class in Pandoc’s markdown.

### Usage

```r
## S3 method for class 'numeric'
pander(x, ...)
```

### Arguments

- **x**: a numeric object
- **...**: ignored parameter
**Description**

Prints a POSIXct object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'POSIXct'
pander(x, ...)
```

**Arguments**

- `x`: a POSIXct object
- `...`: optional parameters passed to `raw_pandoc_date` function

**Description**

Prints a POSIXlt object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'POSIXlt'
pander(x, ...)
```

**Arguments**

- `x`: a POSIXlt object
- `...`: optional parameters passed to `raw_pandoc_date` function
pander.prcomp  

Pander method for prcomp class

Description

Prints a prcomp object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'prcomp'
pander(x, caption = attr(x, "caption"), ...)
```

Arguments

- **x**: a prcomp object
- **caption**: caption (string) to be shown under the table
- **...**: optional parameters passed to raw pandoc.table function

pander.rapport  

Pander method for rapport class

Description

Prints a rapport object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'rapport'
pander(x, ...)
```

Arguments

- **x**: a rapport object
- **...**: ignored parameters
pander.rlm  

*Pander method for rlm class*

---

**Description**

Prints an rlm object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'rlm'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: an rlm object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw pandoc.table function

---

pander.sessionInfo  

*Pander method for sessionInfo class*

---

**Description**

Prints an sessionInfo object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'sessionInfo'
pander(x, locale = TRUE, compact = TRUE, ...)
```

**Arguments**

- `x`: an sessionInfo object
- `locale`: (default:TRUE) if to print locale output
- `compact`: (default:TRUE) if output shoud be compact (ommiting extra line breaks and spaces, inline printing of lists)
- `...`: ignored parameters
pander.smooth.spline  

**Pander method for smooth.spline class**

**Description**

Prints an smooth.spline object in Pandoc's markdown.

**Usage**

```r
## S3 method for class 'smooth.spline'
pander(x, ...)
```

**Arguments**

- `x` an smooth.spline object
- `...` ignored parameters

---

pander.stat.table  

**Pander method for stat.table class**

**Description**

Prints an stat.table object in Pandoc's markdown.

**Usage**

```r
## S3 method for class 'stat.table'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x` an stat.table object
- `caption` caption (string) to be shown under the table
- `...` optional parameters passed to raw pandoc.table function
pander.summary.aov

**Pander method for summary.aov class**

**Description**

Prints a summary.aov object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'summary.aov'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: a summary.aov object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw pandoc.table function

---

pander.summary.aovlist

**Pander method for summary.aovlist class**

**Description**

Prints a summary.aovlist object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'summary.aovlist'
pander(x, caption = attr(x, "caption"), ...)
```

**Arguments**

- `x`: a summary.aovlist object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw pandoc.table function
pander.summary.glm  

**Pander method for summary.glm class**

**Description**

Prints a summary.glm object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'summary.glm'
pander(x, caption = attr(x, "caption"),
       covariate.labels, omit, summary = TRUE, ...)
```

**Arguments**

- `x`: an summary.glm object
- `caption`: caption (string) to be shown under the table
- `covariate.labels`: vector to replace covariate labels in the table
- `omit`: vector of variable to omit for printing in resulting table
- `summary` (default: `TRUE`) if used for summary.lm or lm
- `...`: optional parameters passed to special methods and/or raw pandoc.* functions

**Value**

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

---

pander.summary.lm  

**Pander method for summary.lm class**

**Description**

Prints a summary.lm object in Pandoc’s markdown.

**Usage**

```r
## S3 method for class 'summary.lm'
pander(x, caption = attr(x, "caption"), covariate.labels,
       omit, summary = TRUE, ...)
```
Arguments

- **x**: an `summary.lm` object
- **caption**: caption (string) to be shown under the table
- **covariate.labels**: vector to replace covariate labels in the table
- **omit**: vector of variable to omit for printing in resulting table
- **summary**: (default: `TRUE`) if used for `summary.lm` or `lm`
- **...**: optional parameters passed to special methods and/or raw `pandoc.*` functions

Value

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

---

**pander.summary.prcomp**  
*Pander method for summary.prcomp class*

Description

Prints a `summary.prcomp` object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'summary.prcomp'
pander(x, caption = attr(x, "caption"),
       summary = TRUE, ...)
```

Arguments

- **x**: a `summary.prcomp` object
- **caption**: caption (string) to be shown under the table
- **summary**: (default: `TRUE`) if extended summary should be printed
- **...**: optional parameters passed to raw `pandoc.table` function
pander.survdiff  

Pander method for survdiff class

Description

Prints an survdiff object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'survdiff'
pander(x, caption = attr(x, "caption"), ...)
```

Arguments

- `x`: an survdiff object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw pandoc.table function

pander.survfit  

Pander method for survfit class

Description

Prints an survfit object in Pandoc’s markdown.

Usage

```r
## S3 method for class 'survfit'
pander(x, caption = attr(x, "caption"), scale = 1,
       print.rmean = getOption("survfit.print.rmean"),
       rmean = getOption("survfit.rmean"), ...)
```

Arguments

- `x`: the result of a call to the survfit function.
- `caption`: caption (string) to be shown under the table
- `scale`: a numeric value to rescale the survival time, e.g., if the input data to survfit were in days, scale=365 would scale the printout to years.
- `print.rmean`, `rmean`: Options for computation and display of the restricted mean.
- `...`: optional parameters passed to raw pandoc.table function
## pander.table

**Pander method for table class**

### Description

Prints a table object in Pandoc’s markdown.

### Usage

```r
## S3 method for class 'table'
pander(x, caption = attr(x, "caption"), ...)
```

### Arguments

- `x`: a table object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw `pandoc.table` function

## pander.ts

**Pander method for timeseries class**

### Description

Prints a timeseries object in Pandoc’s markdown.

### Usage

```r
## S3 method for class 'ts'
pander(x, caption = attr(x, "caption"), ...)
```

### Arguments

- `x`: a timeseries object
- `caption`: caption (string) to be shown under the table
- `...`: optional parameters passed to raw `pandoc.table` function
\textit{panderOptions}

\begin{description}
\item[Description] Prints a zoo object in Pandoc’s markdown.
\item[Usage]\begin{verbatim}
## S3 method for class ‘zoo’
pander(x, caption = attr(x, “caption”), ...)
\end{verbatim}
\item[Arguments]\begin{itemize}
\item \texttt{x}: an zoo object
\item \texttt{caption}: caption (string) to be shown under the table
\item ...: optional parameters passed to raw \texttt{pandoc.table} function
\end{itemize}
\end{description}

\textit{panderOptions}

\begin{description}
\item[Description] To list all \texttt{pander} options, just run this function without any parameters provided. To query only one value, pass the first parameter. To set that, use the \texttt{value} parameter too.
\item[Usage]\texttt{panderOptions(o, value)}
\item[Arguments]\begin{itemize}
\item \texttt{o}: option name (string). See below.
\item \texttt{value}: value to assign (optional)
\end{itemize}
\item[Details] The following \texttt{pander} options are available:
\begin{itemize}
\item \texttt{digits}: numeric (default: 2) passed to \texttt{format}
\item \texttt{decimal.mark}: string (default: .) passed to \texttt{format}
\item \texttt{formula.caption.prefix}: string (default: ‘Formula: ’) passed to \texttt{pandoc.formula} to be used as caption prefix. Be sure about what you are doing if changing to other than ‘Formula: ’ or ‘:’.
\end{itemize}
\end{description}
• big.mark: string (default: ") passed to format
• round: numeric (default: Inf) passed to round
• keep.trailing.zeros: boolean (default: FALSE) to show or remove trailing zeros in numbers
• keep.line.breaks: boolean (default: FALSE) to keep or remove line breaks from cells in a table
• date: string (default: "%Y/%m/%d %X") passed to format when printing dates (POSIXct or POSIXt)
• header.style: 'atx' or 'setext' passed to pandoc.header
• list.style: 'bullet', 'ordered' or 'roman' passed to pandoc.list. Please not that this has no effect on pander methods.
• table.style: 'multiline', 'grid', 'simple' or 'rmarkdown' passed to pandoc.table
• table.emphasize.rownames: boolean (default: TRUE) if row names should be highlighted
• table.split.table: numeric passed to pandoc.table and also affects pander methods. This option tells pander where to split too wide tables. The default value (80) suggests the conventional number of characters used in a line, feel free to change (e.g. to Inf to disable this feature) if you are not using a VT100 terminal any more :)
• table.split.cells: numeric or numeric vector (default: 80) passed to pandoc.table and also affects pander methods. This option tells pander where to split too wide cells with line breaks. Numeric vector specifies values for cells separately. Set Inf to disable.
• table.caption.prefix: string (default: 'Table: ') passed to pandoc.table to be used as caption prefix. Be sure about what you are doing if changing to other than 'Table: ' or ':'.
• table.continues: string (default: 'Table continues below') passed to pandoc.table to be used as caption for long (split) without a use defined caption
• table.continues.affix: string (default: '(continued below)') passed to pandoc.table to be used as an affix concatenated to the user defined caption for long (split) tables
• table.alignment.default: string (default: centre) that defines the default alignment of cells. Can be left, right or centre that latter can be also spelled as center.
• table.alignment.rownames: string (default: centre) that defines the alignment of rownames in tables. Can be left, right or centre that latter can be also spelled as center.
• use.hyphenening: boolean (default: FALSE) if try to use hyphenening when splitting large cells according to table.split.cells. Requires roPus package.
• evals.messages: boolean (default: TRUE) passed to evals’ pander method specifying if messages should be rendered
• p.wrap: a string (default: '"') to wrap vector elements passed to p function
• p.sep: a string (default: ', ') with the main separator passed to p function
• p.copula: a string (default: ' and ') with ending separator passed to p function
• plain.ascii: boolean (default: FALSE) to define if output should be in plain ascii or not
• graph.nomargin: boolean (default: TRUE) if trying to keep plots' margins at minimal
• graph.fontfamily: string (default: 'sans') specifying the font family to be used in images. Please note, that using a custom font on Windows requires grDevices::windowsFonts first.
• graph.fontcolor: string (default: 'black') specifying the default font color
panderOptions

- `graph.fontsize`: numeric (default: 12) specifying the base font size in pixels. Main title is rendered with 1.2 and labels with 0.8 multiplier.
- `graph.grid`: boolean (default: TRUE) if a grid should be added to the plot
- `graph.grid.minor`: boolean (default: TRUE) if a minor grid should be also rendered
- `graph.grid.color`: string (default: 'grey') specifying the color of the rendered grid
- `graph.grid.lty`: string (default: 'dashed') specifying the line type of grid
- `graph.borders`: boolean (default: FALSE) if to render a border around of plot (and e.g. around strip)
- `graph.legend.position`: string (default: 'right') specifying the position of the legend: 'top', 'right', 'bottom' or 'left'
- `graph.background`: string (default: 'white') specifying the plots main background's color
- `graph.panel.background`: string (default: 'transparent') specifying the plot's main panel background. Please note, that this option is not supported with base graphics.
- `graph.colors`: character vector of default color palette (defaults to a colorblind theme: http://fly.iam.u-tokyo.ac.jp/color/). Please note that this update work with base plots by appending the col argument to the call if not set.
- `graph.color.rnd`: boolean (default: FALSE) specifying if the palette should be reordered randomly before rendering each plot to get colorful images
- `graph.axis.angle`: numeric (default: 1) specifying the angle of axes' labels. The available options are based on par(les) and sets if the labels should be:
  - 1: parallel to the axis,
  - 2: horizontal,
  - 3: perpendicular to the axis or
  - 4: vertical.
- `graph.symbol`: numeric (default: 1) specifying a symbol (see the pch parameter of par)
- `knitr.auto.asis`: boolean (default: TRUE) if the results of pander should be considered as 'asis' in knitr. Equals to specifying results='asis' in the R chunk, so thus there is no need to do so if set to TRUE.

Note

pander.option is deprecated and is to be removed in future releases.

See Also

evalsOptions

Examples

```r
## Not run:
panderOptions()
panderOptions('digits')
panderOptions('digits', 5)
```
Description

This R5 reference class can hold bunch of elements (text or R objects) from which it tries to create a Pandoc’s markdown text file. Exporting the report to several formats (like: PDF, docx, odt etc. - see Pandoc’s documentation) is also possible, see examples below.

Arguments

... this is an R5 object without any direct params but it should be documented, right?

Methods

export(Class) Returns the result of coercing the object to Class. No effect on the object itself.

Examples

```r
## Not run:
## Initialize a new Pandoc object
myReport <- Pandoc$new()

## Add author, title and date of document
myReport$author <- 'Anonymous'
myReport$title <- 'Demo'

## Or it could be done while initializing
myReport <- Pandoc$new('Anonymous', 'Demo')

## Add some free text
myReport$add.paragraph('Hello there, this is a really short tutorial!')

## Add maybe a header for later stuff
myReport$add.paragraph('# Showing some raw R objects below')

## Adding a short matrix
myReport$add(matrix(5,5))

## Or a table with even # TODO: caption
myReport$add.paragraph('# Hello table:')
myReport$add(table(mtcars$am, mtcars$gear))

## Or a "large" data frame which barely fits on a page
myReport$add(mtcars)

## And a simple linear model with Anova tables
ml <- with(lm(mpg ~ hp + wt), data = mtcars)
myReport$add(ml)
```
myReport$add(anova(ml))
myReport$add(aov(ml))

## And do some principal component analysis at last
myReport$add(prcomp(USArrests))

## Sorry, I did not show how Pandoc deals with plots:
myReport$add(plot(1:10)) # TODO: caption

## Want to see the report? Just print it:
myReport

## Exporting to PDF (default)
myReport$export()

## Or to docx in tempdir():
myReport$format <- 'docx'
myReport$export(tempfile())

## You do not want to see the generated report after generation?
myReport$export(open = FALSE)

## End(Not run)

---

**Pandoc.brew**

**Brew in pandoc format**

---

**Description**

This function behaves just like `brew` except for the `<%...%>` tags, where Pandoc.brew first translate the R object found between the tags to Pandoc's markdown before passing to the `cat` function.

**Usage**

```r
Pandoc.brew(file = stdin(), output = stdout(), convert = FALSE,
open = TRUE, graph.name, graph.dir, graph.hi.res = FALSE, text = NULL,
envir = parent.frame(), append = FALSE, ...)
```

**Arguments**

- `file` file path of the brew template. As this is passed to `readLines`, `file` could be an URL too, but not over SSL (for that latter Rcurl would be needed).
- `output` (optional) file path of the output file
- `convert` string: format of required output document (besides Pandoc's markdown). Pandoc is called if set via `Pandoc.convert` and the converted document could be also opened automatically (see below).
- `open` try to open converted document with operating system's default program
graph.name character string (default to %t when output is set to stdout and paste0(basename(output), '-%n') otherwise) passed to evals. Besides evals's possible tags %i is also available which would be replaced by the chunk number (and optionally an integer which would handle nested brew calls) and %I with the order of the current expression.

graph.dir character string (default to tempdir() when output is set to stdout and dirname(graph.name) otherwise) passed to evals

graph.hi.res render high resolution images of plots? Default is FALSE except for HTML output.

text character vector (treated as the content of the file
environ environment where to brew the template
append should append or rather overwrite (default) the output markdown text file? Please note that this option only affects the markdown file and not the optionally created other formats.

... additional parameters passed to Pandoc.convert

Details

This parser tries to be smart in some ways:

- a block (R commands between the tags) could return any value at any part of the block and there are no restrictions about the number of returned R objects
- plots and images are grabbed in the document, rendered to a png file and pander method would result in a Pandoc's markdown formatted image link (so the image would be shown/included in the exported document). The images are put in plots directory in current getwd() or to the specified output file's directory.
- all warnings/messages and errors are recorded in the blocks and returned in the document as a footnote

Please see my Github page for details (http://rapporter.github.com/pander/#brew-to-pandoc) and examples (http://rapporter.github.com/pander/#examples).

Value

converted file name with full path if convert is set, none otherwise

Note

Only one of the input parameters (file or text) is to be used at once!

References

Examples

```r
## Not run:
text <- paste("# Header", "," ,
'What a lovely list:\n\n\%\=as.list(runif(10))\%\',
'A wide table:\n\n\%\=mtcars[1:3, ]\%\',
'And a nice chart:\n\n\%\=plot(1:10)\%\', sep = ' \n')
Pandoc.brew(text = text)
Pandoc.brew(text = text, output = tempfile(), convert = 'html')
Pandoc.brew(text = text, output = tempfile(), convert = 'pdf')

## pi is awesome
Pandoc.brew(text='\n\n\%\=pi\%\nPi has a lot (<%\=i%) of power: <%=pi^i%> <%%>')

## package bundled examples
Pandoc.brew(system.file('examples/minimal.brew', package='pander'))
Pandoc.brew(system.file('examples/minimal.brew', package='pander'),
  output = tempfile(), convert = 'html')
Pandoc.brew(system.file('examples/short-code-long-report.brew', package='pander'))
Pandoc.brew(system.file('examples/short-code-long-report.brew', package='pander'),
  output = tempfile(), convert = 'html')

## brew returning R objects
str(Pandoc.brew(text='Pi equals to <%=pi%>.
And here are some random data:\n\%\=runif(10)\%\'))

str(Pandoc.brew(text='\n\n\%\=pander\%\nPi is <%=pi%> which is smaller then <%=2%>.
foo\nbar\n\%\=mtcars[1:2, ]\%\'))

str(Pandoc.brew(text='\n\n\%\=pander\%\nPi has a lot (<%\=i%) of power: <%=pi^i%> <%%>'))

## End(Not run)
```

---

**Description**

Calling John MacFarlane’s great program to convert specified file (see `f` parameter below) or character vector see `text` parameter to other formats like HTML, pdf, docx, odt etc.

**Usage**

```r
Pandoc.convert(f, text, format = "html", open = TRUE, options = "",
  footer = TRUE, proc.time, portable.html = TRUE)
```
Arguments

f Pandoc’s markdown format file path. If URL is provided then the generated file’s path is tempfile() but please bear in mind that this way only images with absolute path would shown up in the document.

text Pandoc’s markdown format character vector. Treated as the content of f file - so the f parameter is ignored. The generated file’s path is tempfile().

format required output format. For all possible values here check out Pandoc home-page: http://johnmacfarlane.net/pandoc/

open try to open converted document with operating system’s default program

options optionally passed arguments to Pandoc (instead of pander’s default)

footer add footer to document with meta-information

proc.time optionally passed number in seconds which would be shown in the generated document’s footer

portable.html instead of using local files, rather linking JS/CSS files to an online CDN for portability and including base64-encoded images if converting to HTML without custom options

Value

Converted file’s path.

Note

This function depends on Pandoc which should be pre-installed on user’s machine. See the INSTALL file of the package.

References


Examples

## Not run:
Pandoc.convert(text = c('# Demo', 'with a paragraph'))
Pandoc.convert('http://rapporter.github.io/pander/minimal.md')

## Note: the generated HTML is not showing images with relative path from the above file.
## Based on that `pdf`, `docx` etc. formats would not work! If you want to convert an
## online markdown file to other formats with this function, please pre-process the file
## to have absolute paths instead.

## End(Not run)
pandoc.date.return  

**Description**

Pandoc's markdown date.

**Usage**

\[
\text{pandoc.date.return}(x, \text{inline} = \text{TRUE}, \text{simplified} = \text{FALSE})
\]

**Arguments**

- **x**: date or vector of dates
- **inline**: if to render vector of dates as inline paragraph or not (as list)
- **simplified**: if just add date formatting to vector of dates

**Value**

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

**Examples**

```
pandoc.date(Sys.Date())
pandoc.date(Sys.Date() - 1:10)
pandoc.date(Sys.Date() - 1:10, inline = FALSE)
```

pandoc.emphasis.return

**Emphasis**

**Description**

Pandoc's markdown emphasis format (e.g. `*FOO*`) is added to character string.

**Usage**

\[
\text{pandoc.emphasis.return}(x)
\]

**Arguments**

- **x**: character vector
pandoc.footnote.return

Value
By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

References

See Also
pandoc.strong pandoc.strikeout pandoc.verbatim

Examples
pandoc.footnote('Automatically numbered footnote, right?')
pandoc.formula.return  *Formulas*

**Description**

Pandoc's markdown formula.

**Usage**

```r
pandoc.formula.return(x, text = NULL, max.width = 80, caption,
    add.line.breaks = FALSE)
```

**Arguments**

- `x`  
  formula
- `text`  
  text to be written before result in the same line. Typically used by calls from other functions in the package
- `max.width`  
  maximum width in characters per line
- `caption`  
  caption (string) to be shown under the formula
- `add.line.breaks`  
  if to add 2 line breaks after formula

**Value**

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

**Examples**

```r
pandoc.formula(y ~ x)
pandoc.formula(formula(paste("y ~ ", paste0("x", 1:12, collapse = " + "))))
```

---

pandoc.header.return  *Create header*

**Description**

Creates a (Pandoc's) markdown style header with given level.

**Usage**

```r
pandoc.header.return(x, level = 1, style = c("atx", "setext"))
```
Arguments

- **x** character vector
- **level** integer
- **style** atx or setext type of heading

Value

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

References


Examples

```r
pandoc.header('Foo!', 4)
pandoc.header('Foo!', 2, 'setext')
pandoc.header('Foo **bar**!', 1, 'setext')
```

Description

Creates a Pandoc’s markdown format horizontal line with trailing and leading newlines.

Usage

```r
pandoc.horizontal.rule.return()
```

Value

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

References

Create Pandoc image tags

Description

Creates a Pandoc’s markdown format image hyperlink.

Usage

pandoc.image.return(img, caption = storage$caption)

Arguments

- `img` image path
- `caption` text

Value

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

Note

The `caption` text is read from an internal buffer which defaults to `NULL`. To update that, call `link{set.caption}` before.

References


See Also

`set.caption`

Examples

pandoc.image('foo.png')
pandoc.image('foo.png', 'Nice image, huh?')
**pandoc.indent**  
*Indent text*

**Description**  
Indent all (optionally concatenated) lines of provided text with given level.

**Usage**  
pandoc.indent(x, level = 0)

**Arguments**
- x character vector
- level integer

**Examples**

```
pandoc.indent('FOO', 1)
pandoc.indent(pandoc.table.return(table(mtcars$gear)), 2)  
cat(pandoc.indent(pandoc.table.return(table(mtcars$gear)), 3))
```

---

**pandoc.link.return**  
Create pandoc link Pandoc’s markdown format link.

**Description**  
Create pandoc link Pandoc’s markdown format link.

**Usage**

cat(pandoc.link.return(url, text = url))

**Arguments**
- url hyperlink
- text link text

**Value**

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

**References**

Examples

```r
pandoc.link('http://r-project.org')
pandoc.link('http://r-project.org', 'R')
```

---

**pandoc.list.return**  
*Create a list*

**Description**

Creates a Pandoc’s markdown format list from provided character vector/list.

**Usage**

```r
pandoc.list.return(elements, style = c("bullet", "ordered", "roman"),
                      loose = FALSE, add.line.breaks = TRUE, add.end.of.list = TRUE,
                      indent.level = 0)
```

**Arguments**

- `elements` character vector of strings
- `style` the required style of the list
- `loose` adding a newline between elements
- `add.line.breaks` adding a leading and trailing newline before/after the list
- `add.end.of.list` adding a separator comment after the list
- `indent.level` the level of indent

**Value**

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

**References**


**Examples**

```r
## basic lists
pandoc.list(letters[1:5])
pandoc.list(letters[1:5])
pandoc.list(letters[1:5], 'ordered')
pandoc.list(letters[1:5], 'roman')
pandoc.list(letters[1:5], loose = TRUE)
```
## nested lists

```r
l <- list("First list element",
  rep.int('sub element', 5),
  "Second element",
  list('F', 'B', 'I', c('phone', 'pad', 'talics')))
pandoc.list(l)
pandoc.list(l, loose = TRUE)
pandoc.list(l, 'roman')
```

## complex nested lists

```r
pandoc.list(list('one', as.list(2)))
pandoc.list(list('one', list('two')))  
pandoc.list(list('one', list(2:3)))
```

---

### Description

Pandoc’s markdown paragraph.

### Usage

```r
pandoc.p.return(x)
```

### Arguments

- **x** character vector

### Value

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in `.return`.

### References


### See Also

- `pandoc.emphasis`
- `pandoc.strikeout`
- `pandoc.verbatim`

### Examples

```r
pandoc.p('FOO')
pandoc.p(c('Lorem', 'ipsum', 'lorem ipsum'))
```
**pandoc.strikeout.return**

*Add strikeout*

**Description**

Pandoc’s markdown strikeout format (e.g. `~~FOO~~`) is added to character string.

**Usage**

`pandoc.strikeout.return(x)`

**Arguments**

- `x` character vector

**Value**

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

**References**


**See Also**

- `pandoc.emphasis`
- `pandoc.strong`
- `pandoc.verbatim`

**Examples**

```r
pandoc.strikeout('FOO')
pandoc.strikeout(c('FOO', '~~FOO~~'))
pandoc.strikeout.return('FOO')
```

**pandoc.strong.return**

*Strong emphasis*

**Description**

Pandoc’s markdown strong emphasis format (e.g. `**FOO**`) is added to character string.

**Usage**

`pandoc.strong.return(x)`
Arguments
  x           character vector

Value
  By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

References

See Also
  pandoc.emphasis pandoc.strikeout pandoc.verbatim

Examples
  pandoc.strong(‘FOO’)
  pandoc.strong(c(‘FOO’, ‘**FOO**’))
  pandoc.strong.return(‘FOO’)

pandoc.table.return  Create a table

Description
  Creates a Pandoc’s markdown style table with optional caption and some other tweaks. See ‘Details’ below.

Usage
  pandoc.table.return(t, caption, digits = panderOptions("digits"),
      decimal.mark = panderOptions("decimal.mark"),
      big.mark = panderOptions("big.mark"), round = panderOptions("round"),
      justify, style = c(“multiline”, “grid”, “simple”, “rmarkdown”),
      split.tables = panderOptions("table.split.table"),
      split.cells = panderOptions("table.split.cells"),
      keep.trailing.zeros = panderOptions("keep.trailing.zeros"),
      keep.line.breaks = panderOptions("keep.line.breaks"),
      plain.ascii = panderOptions("plain.ascii"),
      use.hyphenating = panderOptions("use.hyphenating"),
      emphasize.rownames = panderOptions("table.emphasize.rownames"),
      emphasize.rows, emphasize.cols, emphasize.cells, emphasize.strong.rows,
      emphasize.strong.cols, emphasize.strong.cells, …)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>t</code></td>
<td>data frame, matrix or table</td>
</tr>
<tr>
<td><code>caption</code></td>
<td>caption (string) to be shown under the table</td>
</tr>
<tr>
<td><code>digits</code></td>
<td>passed to <code>format</code></td>
</tr>
<tr>
<td><code>decimal.mark</code></td>
<td>passed to <code>format</code></td>
</tr>
<tr>
<td><code>big.mark</code></td>
<td>passed to <code>format</code></td>
</tr>
<tr>
<td><code>round</code></td>
<td>passed to <code>round</code></td>
</tr>
<tr>
<td><code>justify</code></td>
<td>defines alignment in cells passed to <code>format</code>. Can be <code>left</code>, <code>right</code> or <code>centre</code>, which latter can be also spelled as <code>center</code>.Defaults to <code>centre</code></td>
</tr>
<tr>
<td><code>style</code></td>
<td>which Pandoc style to use: <code>simple</code>, <code>multiline</code>, <code>grid</code> or <code>rmarkdown</code></td>
</tr>
<tr>
<td><code>split.tables</code></td>
<td>where to split wide tables to separate tables. The default value (80) suggests the conventional number of characters used in a line, feel free to change (e.g. to <code>Inf</code> to disable this feature) if you are not using a VT100 terminal any more :)</td>
</tr>
<tr>
<td><code>split.cells</code></td>
<td>where to split cells’ text with line breaks. Default to 30, to disable set to <code>Inf</code>. Can be also supplied as a vector, for each cell separately (if length(<code>split.cells</code>) == number of columns + 1, then first value in <code>split.cells</code> if for row names, and others are for columns). Supports relative (percentage) parameters in combination with <code>split.tables</code>.</td>
</tr>
<tr>
<td><code>keep.trailing.zeros</code></td>
<td>to show or remove trailing zeros in numbers on a column basis width</td>
</tr>
<tr>
<td><code>keep.line.breaks</code></td>
<td>(default: <code>FALSE</code>) if to keep or remove line breaks from cells in a table</td>
</tr>
<tr>
<td><code>plain.ascii</code></td>
<td>(default: <code>FALSE</code>) if output should be in plain ascii (without markdown markup) or not</td>
</tr>
<tr>
<td><code>use.hyphenening</code></td>
<td>boolean (default: <code>FALSE</code>) if try to use hyphenening when splitting large cells according to <code>table.split.cells</code>. Requires koRpus package.</td>
</tr>
<tr>
<td><code>emphasize.rownames</code></td>
<td>boolean (default: <code>TRUE</code>) if row names should be highlighted</td>
</tr>
<tr>
<td><code>emphasize.rows</code></td>
<td>a vector for a two dimensional table specifying which rows to emphasize</td>
</tr>
<tr>
<td><code>emphasize.cols</code></td>
<td>a vector for a two dimensional table specifying which cols to emphasize</td>
</tr>
<tr>
<td><code>emphasize.cells</code></td>
<td>a vector for one-dimensional tables or a matrix like structure with two columns for row and column indexes to be emphasized in two-dimensional tables. See e.g. <code>which_HNNNL arr_HNNNL ind_TRUE</code></td>
</tr>
<tr>
<td><code>emphasize.strong.rows</code></td>
<td>see <code>emphasize.rows</code> but in bold</td>
</tr>
<tr>
<td><code>emphasize.strong.cols</code></td>
<td>see <code>emphasize.cols</code> but in bold</td>
</tr>
<tr>
<td><code>emphasize.strong.cells</code></td>
<td>see <code>emphasize.cells</code> but in bold</td>
</tr>
<tr>
<td>...</td>
<td>unsupported extra arguments directly placed into <code>/dev/null</code></td>
</tr>
</tbody>
</table>
Details

This function takes any tabular data as its first argument and will try to make it pretty like: rounding and applying digits and custom decimal.mark to numbers, auto-recognizing if row names should be included, setting alignment of cells and dropping trailing zeros by default.

pandoc.table also tries to split large cells with line breaks or even the whole table to separate parts on demand. Other arguments lets the use to highlight some rows/cells/cells in the table with italic or bold text style.

For more details please see the parameters above and passed arguments of panderoptions.

Value

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call pandoc.table.return instead.

Note

If caption is missing, then the value is first checked in t object's caption attribute and if not found in an internal buffer set by link{set.caption}. justify parameter works similarly, see set.alignment for details.

References


See Also

set.caption, set.alignment

Examples

pandoc.table(mtcars)

## caption
pandoc.table(mtcars, 'Motor Trend Car Road Tests')

## other input/output formats
pandoc.table(mtcars[, 1:3], decimal.mark = ',')
pandoc.table(mtcars[, 1:3], decimal.mark = ',', justify = 'right')
pandoc.table(matrix(sample(1:1000, 25), 5, 5))
pandoc.table(matrix(runif(25), 5, 5))
pandoc.table(matrix(runif(25), 5, 5), digits = 5)
pandoc.table(matrix(runif(25),5,5), round = 1)
pandoc.table(table(mtcars$am))
pandoc.table(table(mtcars$am, mtcars$gear))
pandoc.table(table(state.division, state.region))
pandoc.table(table(state.division, state.region), justify = 'centre')

m <- data.frame(a = c(1, -500, 10320, 23, 77),
                b = runif(5),...
c = c('a', 'bb', 'ccc', 'dddd', 'eeeee')
pandoc.table(m)
pandoc.table(m, justify = c('right', 'left', 'centre'))

# splitting up too wide tables
pandoc.table(mtcars)
pandoc.table(mtcars, caption = 'Only once after the first part!')

# tables with line breaks in cells
# NOTE: line breaks are removed from table content in case keep.line.breaks is set to FALSE
# and added automatically based on "split.cells" parameter!
t <- data.frame(a = c('hundreds\nmouses', '3 cats'), b=c('FOO is nice', 'BAR\nBAR2'))
pandoc.table(t)
pandoc.table(t, split.cells = 5)

# exporting tables in other Pandoc styles
pandoc.table(m)
pandoc.table(m, style = "grid")
pandoc.table(m, style = "simple")
pandoc.table(t, style = "grid")
pandoc.table(t, style = "grid", split.cells = 5)
tryCatch(pandoc.table(t, style = "simple", split.cells = 5),
  error = function(e) 'Yeah, no newline support in simple tables')

# highlight cells
t <- mtcars[1:3, 1:5]
pandoc.table(t$mpg, emphasize.cells = 1)
pandoc.table(t$mpg, emphasize.strong.cells = 1)
pandoc.table(t$mpg, emphasize.cells = 1, emphasize.strong.cells = 1)
pandoc.table(t$mpg, emphasize.cells = 1:2)
pandoc.table(t$mpg, emphasize.strong.cells = 1:2)
pandoc.table(t, emphasize.cells = which(t > 20, arr.ind = TRUE))
pandoc.table(t, emphasize.cells = which(t == 6, arr.ind = TRUE))

# with helpers
emphasize.cols(1)
emphasize.rows(1)
pandoc.table(t)

emphasize.strong.cells(which(t > 20, arr.ind = TRUE))
pandoc.table(t)

### plain.ascii
pandoc.table(mtcars[1:3, 1:3], plain.ascii = TRUE)

### keep.line.breaks
x <- data.frame(a="Pandoc\nPackage")
pandoc.table(x)
pandoc.table(x, keep.line.breaks = TRUE)

### split.cells
x <- data.frame(a = "foo bar", b = "foo bar")
pandoc.table(x, split.cells = 4)
pandoc.table(x, split.cells = 7)
pandoc.title.return

Create title block

Description

Creates a Pandoc’s markdown style title block with optional author, title and date fields.

Usage

pandoc.title.return(author = "", title = "", date = ")

Arguments

author character vector or semicolon delimited list of authors without line break

title character vector of lines of title or multiline string with \n separators

date any string fit in one line

Value

By default this function outputs (see: cat) the result. If you would want to catch the result instead, then call the function ending in .return.

References

Examples

```r
pandoc.title('Tom', 'Render pandoc in R', '2012-05-16')
pandoc.title(c('Tom', 'Jerry'), 'Render pandoc in R', '2012-05-16')
pandoc.title('Tom; Jerry', 'Render pandoc in R', '2012-05-16')
pandoc.title('Tom; Jerry', c('Render', 'pandoc', 'in R'), '2012-05-16')
pandoc.title('Tom; Jerry', 'Render
 pandoc
 in R', '2012-05-16')

## missing fields

pandoc.title('Tom; Jerry', 'Render pandoc in R')
pandoc.title('Tom; Jerry')
pandoc.title(title = 'Render pandoc in R', date = '2012-05-16')
```

---

## Add verbatim

### Description

Pandoc's markdown verbatim format (e.g. `FOO`) is added to character string.

### Usage

```r
pandoc.verbatim.return(x, style = c("inline", "indent", "delim"),
                        attrs = "")
```

### Arguments

- **x**: character vector
- **style**: show code inline or in a separate (indented or delimited) block
- **attrs**: (optionally) pass ID, classes and any attribute to the delimited block

### Value

By default this function outputs (see: `cat`) the result. If you would want to catch the result instead, then call the function ending in `.return`.

### References


### See Also

pandoc.emphasis pandoc.strikeout pandoc.strong
redraw.recordedplot  Redraws plot saved in file

Description

This function is a wrapper around `redrawPlot`.

Usage

redraw.recordedplot(file)

Arguments

file  path and name of an rds file containing a plot object to be redrawn

References

Thanks to Jeroen Ooms http://permalink.gmane.org/gmane.comp.lang.r.devel/29897, JJ Allaire https://github.com/rstudio/rstudio/commit/eb5f6f1db4717132c2ff11f068ffa6e8b2a5f0b, and Gabriel Becker.

See Also

evals
redrawPlot  
Redraw a recordedplot, grid, trellis, or ggplot2 plot.

Description
This function redraws the plot represented by recPlot. It can redraw grid/trellis/ggplot2/etc plots, as well as recordedplot objects. For recordedplot objects it acts as a wrapper around replayPlot with memory tweaks to fix native symbol address errors when the recordedplot was loaded from an rda/rds file.

Usage
redrawPlot(recPlot)

Arguments
recPlot  the plot object to redraw

References
Thanks to Jeroen Ooms http://permalink.gmane.org/gmane.comp.lang.r.devel/29897, JJ Allaire https://github.com/rstudio/rstudio/commit/eb5f6f1db4717132c2ff111f068ffa6e8b2a5f0b, and Gabriel Becker.

See Also
redraw.recordedplot

remove.extra.newlines  Remove more then two joined newlines

Description
Remove more then two joined newlines

Usage
remove.extra.newlines(x)

Arguments
x  character vector

Examples
remove.extra.newlines(c(‘\n\n\n’, ‘\n\n’, ‘\n’))
repChar  

Description
Repeating a string n times and returning a concatenated character vector.

Usage
repChar(x, n, sep = "")

Arguments
x    string to repeat
n    integer
sep   separator between repetitions

Value
character vector

set.alignment  

Description
This is a helper function to update the alignment (justify parameter of pandoc.table) of the returning table. Possible values are: centre or center, right, left.

Usage
set.alignment(default = "centre", row.names = "right", permanent = FALSE)

Arguments
default   character vector which length equals to one (would be repeated n times) or n - where n equals to the number of columns in the following table
row.names   string holding the alignment of the (optional) row names
permanent   (default FALSE) if alignment is permanent (for all future tables) or not
set.caption | Adds caption in current block

Description
This is a helper function to add a caption to the returning image/table.

Usage
set.caption(x, permanent = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>string</td>
<td>string to be split. Works only with one string. Non-string arguments and multi-dimensional arguments are returned unchanged</td>
</tr>
<tr>
<td>permanent</td>
<td>(default FALSE)</td>
<td>if caption is permanent (for all future tables) or not</td>
</tr>
</tbody>
</table>

splitLine | Split line with line breaks depending on max.width

Description
This is a helper function to insert line breaks depending on (split.cells parameter of pandoc.table) of the returning table.

Usage
splitLine(x, max.width = panderOptions("table.split.cells"),
use.hyphenating = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>string</td>
<td>string to be split. Works only with one string. Non-string arguments and multi-dimensional arguments are returned unchanged</td>
</tr>
<tr>
<td>max.width</td>
<td>default integer value specifying max number of characters between line breaks</td>
<td></td>
</tr>
<tr>
<td>use.hyphenating</td>
<td>(default: FALSE)</td>
<td>if try to use hyphenation when splitting large cells according to table.split.cells. Requires koRpus package.</td>
</tr>
</tbody>
</table>

Value
character string with line breaks

Examples

splitLine("foo bar", 6)
splitLine("foo bar", 7)
splitLine("Pandoc Package", 3, TRUE)
trim.spaces

Trim leading and trailing spaces

Description
Trim leading and trailing spaces

Usage
trim.spaces(x)

Arguments
x character vector

Value
character vector

See Also
trim.space in rapport package

wrap

Wrap Vector Elements

Description
Wraps vector elements with string provided in wrap argument.

Usage
wrap(x, wrap = "\")

Arguments
x a vector to wrap
wrap a string to wrap around vector elements

Value
a string with wrapped elements

Author(s)
Aleksandar Blagotic
References

This function was moved from rapport package: http://rapport-package.info/.

Examples

```r
## Not run:
wrap("foobar")
wrap(c("fee", "fi", "foo", "fam"), ")

## End(Not run)
```
Index

add.blank.lines, 4
add.significance.stars, 4

cache.off, 5
cache.on(cache.off), 5

emphasize.cells(emphasize.rows), 5
emphasize.cols(emphasize.rows), 5
emphasize.rows, 5
emphasize.strong.cells
(emphasize.rows), 5
emphasize.strong.cols(emphasize.rows), 5
emphasize.strong.rows(emphasize.rows), 5

evals, 5, 7, 8, 10, 11, 15–17, 51, 69
evals.option(evalsOptions), 15
evalsOptions, 5, 11, 15, 48

has.rownames, 17

openFileInOS, 18

p, 18, 18
pander, 5, 19
pander.anova, 21
pander.aov, 22
pander.aovlist, 22
pander.call, 23
pander.cast.df, 23
pander.character, 24
pander.clogit, 24
pander.coxph, 25
pander.CrossTable, 25
pander.data.frame, 26
pander.Date, 26
pander.default, 27
pander.density, 27
pander.describe, 28
pander.evals, 28

pander.factor, 29
pander.formula, 29
pander.ftable, 30
pander.function, 30
pander.glm, 31
pander.htest, 31
pander.image, 32
pander.list, 32
pander.lm, 33
pander.lme, 33
pander.logical, 34
pander.matrix, 34
pander.microbenchmark, 35
pander.mtable, 35
pander.NULL, 36
pander.numeric, 36
pander.option(panderOptions), 46
pander.POSIXct, 37
pander.POSIXlt, 37
pander.prcomp, 38
pander.rapport, 38
pander.rlm, 39
pander.sessionInfo, 39
pander.smooth.spline, 40
pander.stat.table, 40
pander.summary.aov, 41
pander.summary.aovlist, 41
pander.summary.glm, 42
pander.summary.lm, 42
pander.summary.prcomp, 43
pander.survdiff, 44
pander.survfit, 44
pander.table, 45
pander.ts, 45
pander.zoo, 46
panderOptions, 17, 46, 65
Pandoc (Pandoc-class), 49
Pandoc-class, 49
Pandoc.brew, 9, 50
Pandoc.convert, 51, 52
pandoc.date (pandoc.date.return), 54
pandoc.date.return, 54
pandoc.emphasis, 61–63, 68
pandoc.emphasis
  (pandoc.emphasis.return), 54
pandoc.emphasis.return, 54
pandoc.footnote
  (pandoc.footnote.return), 55
pandoc.footnote.return, 55
pandoc.formula, 46
pandoc.formula (pandoc.formula.return), 56
pandoc.formula.return, 56
pandoc.header, 47
pandoc.header (pandoc.header.return), 56
pandoc.header.return, 56
pandoc.horizontal.rule
  (pandoc.horizontal.rule.return), 57
pandoc.horizontal.rule.return, 57
pandoc.image (pandoc.image.return), 58
pandoc.image.return, 58
pandoc.indent, 59
pandoc.link (pandoc.link.return), 59
pandoc.link.return, 59
pandoc.list, 47
pandoc.list (pandoc.list.return), 60
pandoc.list.return, 60
pandoc.p (pandoc.p.return), 61
pandoc.p.return, 61
pandoc.strikeout, 55, 61, 63, 68
pandoc.strikeout
  (pandoc.strikeout.return), 62
pandoc.strikeout.return, 62
pandoc.strong, 55, 62, 68
pandoc.strong (pandoc.strong.return), 62
pandoc.strong.return, 62
pandoc.table, 5, 47
pandoc.table (pandoc.table.return), 63
pandoc.table.return, 63
pandoc.title (pandoc.title.return), 67
pandoc.title.return, 67
pandoc.verbatim, 55, 61–63
pandoc.verbatim
  (pandoc.verbatim.return), 68
pandoc.verbatim.return, 68
redraw.recordedplot, 69, 70
redrawPlot, 70
remove.extra.newlines, 70
repChar, 71
set.alignment, 65, 71
set.caption, 58, 65, 72
splitLine, 72
tempfile, 9, 16
trim.spaces, 73
wrap, 73