Package 'qwraps2'

September 27, 2016

Title Quick Wraps 2

Version 0.2.2

Description A collection of (wrapper) functions the creator found useful for quickly placing data summaries and formatted regression results into '.Rnw' or '.Rmd' files. Functions for generating commonly used graphics, such as receiver operating curves or Bland-Altman plots, are also provided by 'qwraps2'. 'qwraps2' is a updated version of a package 'qwraps'. The original version 'qwraps' was never submitted to CRAN but can be found at <https://github.com/dewittpe/qwraps/>. The implementation and limited scope of the functions within 'qwraps2' <https://github.com/dewittpe/qwraps2/> is fundamentally different from 'qwraps'.

Depends R (>= 3.0.2)

License GPL-2

URL https://github.com/dewittpe/qwraps2/

LazyData true

Imports dplyr, ggplot2, knitr, Rcpp

Suggests survival, testthat, covr, rmarkdown

RoxygenNote 5.0.1

LinkingTo Rcpp (>= 0.11.0)

VignetteBuilder knitr

NeedsCompilation yes

Author Peter DeWitt [aut, cre],
Tell Bennett [ctb]

Maintainer Peter DeWitt <dewittpe@gmail.com>

Repository CRAN

Date/Publication 2016-09-27 16:39:55

R topics documented:

`confusion_matrix` *Confusion Matrices (Contingency Tables)*

Description

Construction of confusion matrices, accuracy, sensitivity, specificity, confidence intervals (Wilson's method and (optional bootstrapping)).

Usage

```
confusion_matrix(x, ...)

## Default S3 method:
confusion_matrix(x, y, positive, boot = FALSE,
  boot_samples = 1000L, alpha = 0.05, ...)

## S3 method for class 'formula'
confusion_matrix(formula, data = parent.frame(), positive,
  boot = FALSE, boot_samples = 1000L, alpha = 0.05, ...)

is.confusion_matrix(x)

## S3 method for class 'confusion_matrix'
print(x, ...)
```

Arguments

x	prediction condition vector, a two level factor variable or a variable that can be converted to one.
...	not currently used
y	True Condition vector with the same possible values as x.
positive	the level of x and y which is the positive outcome. If missing the first level of factor(y) will be used as the positive level.
boot	boolean, should bootstrapped confidence intervals for the sensitivity and specificity be computed? Defaults to FALSE.
boot_samples	number of bootstrapping sample to generate, defaults to 1000L. Ignored if `boot == FALSE`.
alpha	100(1-alpha) sensitivity. Ignored if `boot == FALSE`.
formula	column (known) ~ row (test) for building the confusion matrix
data	environment containing the variables listed in the formula

Details

Sensitivity and Specificity: For the sensitivity and specificity function we expect the 2-by-2 confusion matrix (contingency table) to be of the form:

		True Condition +	Condition -
Predicted Condition	+	TP	FP
Predicted Condition	-	FN	TN

where

- FN: False Negative, and
- FP: False Positive,
- TN: True Negative,
- TP: True Positive.

Recall:

- sensitivity = TP / (TP + FN)
- specificity = TN / (TN + FP)
- positive predictive value (PPV) = TP / (TP + FP)
- negative predictive value (NPV) = TN / (TN + FN)

Value

The sensitivity and specificity functions return numeric values. `confusion_matrix` returns a list with elements:

- tab the confusion matrix,
- stats a matrix of summary statistics and confidence intervals.

Examples

```
## Example taken from caret::confusionMatrix

lvs <- c("normal", "abnormal")
truth <- factor(rep(lvs, times = c(86, 258)),
                levels = rev(lvs))
pred <- factor(c(rep(lvs, times = c(54, 32)),
                 rep(lvs, times = c(27, 231))),
               levels = rev(lvs))

confusion_matrix(pred, truth)
confusion_matrix(pred, truth)$stats

confusion_matrix(pred, truth, boot = TRUE)
confusion_matrix(pred, truth, positive = "normal", boot = TRUE)

# Using formulas
test_data <- data.frame(xyz = pred, yyy = truth)
confusion_matrix(yyy ~ xyz, test_data)
confusion_matrix(yyy ~ xyz, test_data, positive = "normal")
```

`extract_fstat` *Extract Summary stats from regression objects*

Description

A collection of functions for extracting summary statistics and reporting regression results from `lm`, `glm` and other regression objects.

Usage

```
extract_fstat(x)

extract_fpvalue(x)

## S3 method for class 'lm'
extract_fpvalue(x)
```

Arguments

`x`	a `lm` object

Details

TO DO

Value

a character vector of the formatted numbers

formatted p-value from the F-test

See Also

lm

Examples

```
# TODO
```

`frmt` *Format Wrappers*

Description

Functions for formating Numeric values for consistent display in reports.

Usage

```
frmt(x, digits = getOption("qwraps2_frmt_digits", 2))

frmtp(x, style = getOption("qwraps2_journal", "default"),
  digits = getOption("qwraps2_frmtp_digits", 4),
  markup = getOption("qwraps2_markup", "latex"),
  case = getOption("qwraps2_frmtp_case", "upper"),
  leading0 = getOption("qwraps2_frmtp_leading0", TRUE))

frmtci(x, est = 1, lcl = 2, ucl = 3, format = "est (lcl, ucl)",
  show_level = FALSE, ...)
```

Arguments

`x`	a vector of numbers or a numeric matrix to format.
`digits`	number of digits, including trailing zeros, to the right of the decimal point. This option is ignored if `is.integer(x) == TRUE`).
`style`	a character string indicating a specific journal requirements for p-value formatting.
`markup`	a character string indicating if the output should be latex or markup.
`case`	a character string indicating if the output should be upper case or lower case.
`leading0`	boolean, whether or not the p-value should be reported as 0.0123 (TRUE, default), or .0123 (FALSE).

est	the numeric index of the vector element or the matrix column containing the point estimate.
lcl	the numeric index of the vector element or the matrix column containing the lower confidence limit.
ucl	the numeric index of the vector element or the matrix column containing the upper confidence limit.
format	a string with "est" "lcl", and "ucl" to denote the location of the estimate, lower confidence limit, and upper confidence limit for the formated string. Defaults to "est (lcl, ucl)".
show_level	defualts to FALSE. If TRUE and format is the default, then "100*(1-options()$qwraps2_alpha) parenthesis and the lcl. If set to a string, then the given string will be placed between the left parenthesis and the lcl. If the format is not the default, then this argument is ignored.
...	args passed to frmt

Details

'frmt' is really just a wrapper for the formatC.

'frmtp' formats P-values per journal requirements. As I work on papers aimed at different journals, the formatting functions will be extended to match.

Default settings are controlled through the function arguments but should be set via options().

Default settings report the P-value exactly if P > getOptions("qwraps2_frmtp_digits", 4) and reports P < 10^-(getOptions("qwraps2_frmtp_digits", 2)) otherwise. By the leading zero is controlled via getOptions("qwraps2_frmtp_leading0", TRUE) and a upper or lower case P is controlled by getOptions("qwraps2_frmtp_case", "upper"). These options are ignored if style != "default".

Journals with predefined P-value formatting are noted in the **qwraps2** documentation.

'frmtci' takes a matrix, or data.frame, with a point estimate and the lcl and ucl and formats a string for reporting. est (lcl, ucl) is the default. The confidence level can be added to the string, e.g., "est (95 format.

'frmtcip' expects four values, est, lcl, ucl, and p-value. The resulting sting will be of the form "est (lcl, ucl; p-value)".

Value

a character vector of the formatted numbers

See Also

formatC

Examples

```
# Formatting numbers
integers <- c(1234L, 9861230L)
```

```
numbers  <- c(1234,  9861230)
frmt(integers)  # no decimal point
frmt(numbers)   # decimal point and zeros to the right

numbers <- c(0.1234, 0.1, 1234.4321, 0.365, 0.375)
frmt(numbers)

# Formatting p-values
ps <- c(0.2, 0.001, 0.00092, 0.047, 0.034781, 0.0000872, 0.787, 0.05, 0.043)
# LaTeX is the default markup language
cbind("raw"      = ps,
      "default"  = frmtp(ps),
      "3lower"   = frmtp(ps, digits = 3, case = "lower"),
      "PediDent" = frmtp(ps, style = "pediatric_dentistry"))

# Using markdown
cbind("raw"      = ps,
      "default"  = frmtp(ps, markup = "markdown"),
      "3lower"   = frmtp(ps, digits = 3, case = "lower", markup = "markdown"),
      "PediDent" = frmtp(ps, style = "pediatric_dentistry", markup = "markdown"))

# Formatting the point estimate and confidence interval
# for a set of three values
temp <- c(a = 1.23, b = .32, CC = 1.78)
frmtci(temp)
frmtci(temp, show_level = TRUE)

# note that the show_level will be ignored in the following
frmtci(temp, format = "est ***lcl, ucl***", show_level = TRUE)

# show_level as a character
frmtci(temp, show_level = "confidence between: ")

# For a matrix: the numbers in this example don't mean anything, but the
# formatting should.
temp2 <- matrix(rnorm(12), nrow = 4,
                dimnames = list(c("A", "B", "C", "D"), c("EST", "LOW", "HIGH")))
temp2
frmtci(temp2)
```

`ggplot2_extract_legend` *ggplot2 tools*

Description

A few handy tools for working with ggplot2.

Usage

```
ggplot2_extract_legend(x)
```

Arguments

x	a ggplot

Details

The ggplot2_extract_legend function returns a list with the first element being the legend and the second the original plot with the legend omitted.

Value

a list with each element a ggplot

Examples

```
# a simple plot
my_plot <-
  ggplot2::ggplot(mtcars) +
  ggplot2::aes(x = wt, y = mpg, color = wt, shape = factor(cyl)) +
  ggplot2::geom_point()

my_plot

# extract the legend.  the return object is a list with two elements, the first
# element is the legend, the second is the original plot sans legend.
temp <- ggplot2_extract_legend(my_plot)

# view just the legend.  This can be done via a call to the object or using
# plot or print.
temp
plot(temp[[1]])

# the original plot without the legened
plot(temp[[2]])
```

---

logit *logit and inverse logit functions*

Description

transform x either via the logit, or inverse logit.

Usage

```
logit(x)

invlogit(x)
```

Arguments

x	a numeric vector

Details

Why are these functions are not part of base R? I don't have the answer to that question. So here are functions for easy of use.

`mean_ci` *Means and Confidence Intervals*

Description

A function for calculating and formatting means and confidence interval.

Usage

```
mean_ci(x, na_rm = FALSE, transform, alpha = getOption("qwraps2_alpha",
  0.05), qdist = stats::qnorm, qdist.args = list())

## S3 method for class 'qwraps2_mean_ci'
print(x, ...)
```

Arguments

x	a numeric vector
na_rm	if true, omit NA values
transform	function transform to the mean and the confidence limits. See Details.
alpha	defaults to `getOption('qwraps2_alpha', 0.05)`. The symmetric 100(1-alpha)% CI will be determined.
qdist	defaults to `qnorm`. use `qt` for a Student t intervals.
qdist.args	list of arguments passed to `qdist`
...	arguments passed to `frmtci`.

Details

Given a numeric vector, `mean_ci` will return a vector with the mean, LCL, and UCL. Using `frmtci` will be helpfull for reporting the results in print.

The `transform` arguement allows the use to transform the data. A common occurance of using `mean_ci(log(x), transform = exp)` will return the geometric mean and confidence interval for x.

Value

a vector with the mean, lower confidence limit (LCL), and the upper confidence limit (UCL).

See Also

frmtci

Examples

```
# using the standard normal for the CI
mean_ci(mtcars$mpg)

# print it nicely
qwraps2::frmtci(mean_ci(mtcars$mpg))
qwraps2::frmtci(mean_ci(mtcars$mpg), show_level = TRUE)
qwraps2::frmtci(mean_ci(mtcars$mpg, alpha = 0.01), show_level = TRUE)

# Compare to the ci that comes form t.test
t.test(mtcars$mpg)
t.test(mtcars$mpg)$conf.int
mean_ci(mtcars$mpg, qdist = stats::qt, qdist.args = list(df = 31))

# geometric version
mean_ci(log(mtcars$mpg), transform = exp, qdist = stats::qt, qdist.args = list(df = 31))
```

---

mean_sd *Mean and Standard deviation*

Description

A function for calculating and formatting means and standard deviations.

Usage

```
mean_sd(x, digits = getOption("qwraps2_frmt_digits", 2), na_rm = FALSE,
  show_n = "ifNA", denote_sd = "pm", markup = getOption("qwraps2_markup",
  "latex"))

gmean_sd(x, logged = FALSE, digits = getOption("qwraps2_frmt_digits", 2),
  na_rm = FALSE, show_n = "ifNA", denote_sd = "pm",
  markup = getOption("qwraps2_markup", "latex"))
```

Arguments

x	a numeric vector
digits	digits to the right of the decimal point to return in the percentage estimate.
na_rm	if true, omit NA values
show_n	defaults to "ifNA". Other options are "always" or "never".
denote_sd	a character string set to either "pm" or "paren" for reporting 'mean $\pm$ sd' or 'mean (sd)'

markup	latex or markdown
logged	if TRUE the x is assumed to already by log transformed. If FALSE the data will be transformed.

Details

Given a numeric vector, mean_sd will return a character string with the mean and standard deviation. Formating of the output will be extended in future versions.

gmean_sd returns the geometric mean and geometric standard deviation.

Value

a character vector of the formatted values

Examples

```
set.seed(42)
x <- rnorm(1000, 3, 4)
mean(x)
sd(x)
mean_sd(x)
mean_sd(x, show_n = "always")
mean_sd(x, show_n = "always", denote_sd = "paren")

x[187] <- NA
mean_sd(x, na_rm = TRUE)
```

---

median_iqr *Median and Inner Quartile Range*

Description

A function for calculating and formatting the median and inner quartile range of a data vector. #' @details Given a numeric vector, median_iqr will return a character string with the median and IQR. Formating of the output will be extended in future versions.

Usage

```
median_iqr(x, digits = getOption("qwraps2_frmt_digits", 2), na_rm = FALSE,
  show_n = "ifNA", markup = getOption("qwraps2_markup", "latex"))
```

Arguments

x	a numeric vector
digits	digits to the right of the decimal point to return.
na_rm	if true, omit NA values
show_n	defaults to "ifNA". Other options are "always" or "never".
markup	latex or markdown

Value

a character vector of the formatted values

Examples

```
set.seed(42)
x <- rnorm(1000, 3, 4)
median(x)
quantile(x, probs = c(1, 3)/4)
median_iqr(x)
median_iqr(x, show_n = "always")

x[187] <- NA
# median_iqr(x) ## Will error
median_iqr(x, na_rm = TRUE)
```

---

n_perc *Count and Percentage*

Description

A function for calculating and formatting counts and percentages.

Usage

```
n_perc(x, digits = getOption("qwraps2_frmt_digits", 2), na_rm = FALSE,
  show_denom = "ifNA", show_symbol = TRUE,
  markup = getOption("qwraps2_markup", "latex"))

perc_n(x, digits = getOption("qwraps2_frmt_digits", 2), na_rm = FALSE,
  show_denom = "ifNA", markup = getOption("qwraps2_markup", "latex"))

n_perc0(x, digits = 0, na_rm = FALSE, show_denom = "never",
  show_symbol = FALSE, markup = getOption("qwraps2_markup", "latex"))
```

Arguments

x	a 0:1 or boolean vector
digits	digits to the right of the decimal point to return in the percentage estimate.
na_rm	if true, omit NA values
show_denom	defaults to "ifNA". Other options are "always" or "never".
show_symbol	if TRUE (default) the percent symbol is shown, else it is supressed.
markup	latex or markdown

Details

Default behavior will return the count of successes and the percentage as "N (pp can be controlled by setting `na.rm = TRUE`. In this case, the number of non-missing values will be reported by default. Omission of the non-missing values can be controlled by setting `show_denom = "never"`.

The function n_perc0 uses a set of default arguments which may be advantageous for use in building tables.

Value

a character vector of the formatted values

Examples

```
n_perc(c(0, 1,1, 1, 0, 0), show_denom = "always")
n_perc(c(0, 1,1, 1, 0, 0, NA), na_rm = TRUE)

n_perc(mtcars$cyl == 6)

set.seed(42)
x <- rbinom(4269, 1, 0.314)
n_perc(x)
n_perc(x, show_denom = "always")
n_perc(x, show_symbol = FALSE)

# n_perc0 examples
n_perc0(c(0, 1,1, 1, 0, 0))
n_perc0(mtcars$cyl == 6)
```

---

qable *Qable: an extended verion of knitr::kable*

Description

Create a simple table via kable with row groups and rownames similar to those of `hmisc::latex` or `htmlTable::htmlTable`.

Usage

```
qable(x, rgroup, rnames = rownames(x), cnames = colnames(x),
  markup = getOption("qwraps2_markup", "latex"), ...)
```

Arguments

x	matrix or data.frame to be turned into a qable
rgroup	a named numeric vector with the name of the row group and the number of rows within the group. sum(rowgroup) == nrow(x).
rnames	a character vector of the row names
cnames	column names
markup	the markup language to use, passed to the format argument of knitr::kable.
...	additional arguments passed to knitr::kable

Details

qable is used as the printing method for qwraps2_summary_table objects. Check the vignettes for examples on building data summary tables.

Value

a character vector of the formatted numbers

See Also

kable summary_table, for

Examples

```
data(mtcars)
qable(mtcars)
qable(mtcars, markup = "markdown")

# by make
make <- sub("^(\\w+)\\s?(.*)$", "\\1", rownames(mtcars))
make <- c(table(make))

qable(mtcars[sort(rownames(mtcars)), ], rgroup = make)
qable(mtcars[sort(rownames(mtcars)), ], rgroup = make, markup = "markdown")

# define your own column names
qable(mtcars[sort(rownames(mtcars)), ],
      rgroup = make,
      cnames = toupper(colnames(mtcars)),
      markup = "markdown")
```

qacf *Autocorrelation plot*

Description

TO BE CONSTRUCTED

Usage

```
qacf(x, ...)
```

Arguments

x	object
...	Other arguments passed to stats::acf

Details

TO DO

Value

a ggplot.

qblandaltman *Bland Altman Plots*

Description

Construct and plot a Bland Altman plot in ggplot2.

Usage

```
qblandaltman(.data, alpha = getOption("qwraps2_alpha", 0.05),
  generate_data = TRUE)

qblandaltman_build_data_frame(.data, alpha = getOption("qwraps2_alpha", 0.05))
```

Arguments

.data	a data.frame with two columns. If a data.frame with more than two columns is used only the first two columns will be used.
alpha	(Defaults to 0.05) place (1 - alpha)*100 place on the plot.
generate_data	logical, defaults to TRUE. If TRUE, then the call to qblandaltman_build_data_frame is done automatically for you. If FALSE, then you should explicitly call qblandaltman_build_data_fra before calling qblandaltman.

Details

Providing a `data.frame` with two columns, the function returns a ggplot version of a Bland Altman plot with the specified confidence intervals.

Two ways to call the plotting function. If you submit a `data.frame` `qblandaltman` then the data needed to produce the Bland Altman plot is automatically generated by a call to `qblandaltman_build_data_frame`. Alternatively, you may call `qblandaltman_build_data_frame` directly and then call `qblandaltman`. This might be helpful if you are putting multiple Bland Altman plots together into one ggplot object. See Examples.

Value

a ggplot. Minimal aesthetics have been used so that the user may modify the graphic as desired with ease.

References

Altman, Douglas G., and J. Martin Bland. "Measurement in medicine: the analysis of method comparison studies." The statistician (1983): 307-317.

Bland, J. Martin, and DouglasG Altman. "Statistical methods for assessing agreement between two methods of clinical measurement." The lancet 327, no. 8476 (1986): 307-310.

Examples

```
## Not run:
# load ggplot2 and the diamonds data set
library(ggplot2)
data(diamonds, package = "ggplot2")

# compare a simple regression to random noise
dat <-
  data.frame(fitted(lm(price ~ poly(carat, 4), data = diamonds)),  # fitted values
             diamonds$price + rnorm(nrow(diamonds), sd = 0.2),     # observed with noise
             pi)                                                   # extra column
qblandaltman(dat)

# simple example
dat <- data.frame(eval1 = rpois(100, 3), eval2 = rpois(100, 3.4))
qblandaltman(dat)

last_plot() + theme_bw()

# Two plots in one ggplot object
set.seed(42)
dat1 <- data.frame(eval1 = rnorm(100), eval2 = rt(100, df = 1))
dat2 <- data.frame(eval1 = rpois(50, 3), eval2 = rpois(50, 4))

# individual plots
qblandaltman(dat1)
qblandaltman(dat2)
```

```
# combined plots
dat <- rbind(cbind(set = "rnorm", qblandaltman_build_data_frame(dat1)),
             cbind(set = "rpois", qblandaltman_build_data_frame(dat2)))
qblandaltman(dat, generate_data = FALSE) + facet_wrap( ~ set)

## End(Not run)
```

---

qkmplot *Kaplan-Meier Plot*

Description

A ggplot2 version of a Kaplan-Meier Plot

Usage

```
qkmplot(x, conf_int = FALSE, ...)

qkmplot_bulid_data_frame(x)
```

Arguments

x	object
conf_int	logical if TRUE show the CI
...	Other arguments passed to survival::plot.survfit

Details

Functions to build, explicitly or implicitly, data.frames and then creating a ggplot2 KM plot.

Value

a ggplot.

Examples

```
require(survival)
leukemia.surv <- survival::survfit(survival::Surv(time, status) ~ x, data = survival::aml)
survival:::plot.survfit(leukemia.surv, conf.int = TRUE, lty = 2:3, col = 1:2)

qkmplot(leukemia.surv, conf_int = TRUE)
```

qroc *Receiver Operator Curves*

Description

Construction of ROC curves.

Usage

```
qroc(x, ...)

qroc_build_data_frame(fit, n_threshold = 200)

auc(.data)
```

Arguments

x	a glm fit or data.frame generated by qroc_build_data_frame.
...	Ignored
fit	a glm fit with family = binomial().
n_threshold	number of thresholds to test against.
.data	a data.frame generated by qroc_build_data_frame.

Details

Given a glm fit with family = "binomial" (either a log-link or logit-link should be fine, a data set will be constructed and ROC plots generated.

The area under the curve (AUC) is determined by a trapezoid approximation.

Value

a ggplot. Minimal aesthetics have been used so that the user may modify the graphic as desired with ease.

AUC for the data set generated by

Examples

```
## Not run:
# load ggplot2 and the diamonds data set
library(ggplot2)
data(diamonds, package = "ggplot2")

# Create two logistic regression models
fit1 <- glm(I(price > 2800) ~ cut * color, data = diamonds, family = binomial())
fit2 <- glm(I(price > 2800) ~ cut + color + clarity, data = diamonds, family = binomial())
```

```
# Easiest way to get an ROC plot:
qroc(fit1)
qroc(fit2)

# Create two data sets, this will also let you get the AUC out
data1 <- qroc_build_data_frame(fit1)
data2 <- qroc_build_data_frame(fit2)

auc(data1)
auc(data2)

# Plotting the ROC from the data set can be done too
qroc(data1)

# Add the AUC value to the plot title
qroc(data2) + ggtitle(paste("Fit 2\nAUC =", round(auc(data2), 2)))

# build a data set for plotting to ROCs on one plot
plot_data <- rbind(cbind(Model = "fit1", data1),
                   cbind(Model = "fit2", data2))
qroc(plot_data) + aes(color = Model)

# with AUC in the legend
plot_data <- rbind(cbind(Model = paste("Fit1\nauc =", round(auc(data1), 3)), data1),
                   cbind(Model = paste("Fit2\nauc =", round(auc(data2), 3)), data2))
qroc(plot_data) +
  theme_bw() +
  aes(color = Model, linetype = Model) +
  theme(legend.position   = "bottom",
        legend.text.align = 0.5)

## End(Not run)
```

`qwraps2` *A collection of wrapper functions aimed at for aiding the authoring of reproducible reports.*

Description

qwraps2 is a collection of helpful functions when working on a varied collection of different analysis reports. There are two types of functions, helpful data summary functions, formatting results from regression models, and **ggplot2** wrappers.

Details

Several wrappers for **ggplot2** style graphics, such as ROC, AUC, Bland-Altman, and KM plots are provided. Named as `qroc`, `qacf`, `qblandaltman` and `qkmplot` to pay homage to `qplot` form **ggplot2** and the standard names for such plots.

Other functions are used to quickly generate meaningful character strings for outputting results in .Rnw, .Rmd, or other similar functions.

Options

There are several options which can be set via `options` and will be used via `getOption`. The following lists, in alphabetical order the different options which are available and what they control.

- `getOptions("qwraps2_alpha", 0.05)` significance level, used for generating `(1 - getOptions("qwraps2_alpha", 0.05))` confidence intervals, and determining significance for p-value < `getOptions("qwraps2_alpha", 0.05)`.
- `getOptions("qwraps2_frmt_digits", 2)` Number of digits to the right of the decimal point for any value other than p-values.
- `getOptions("qwraps2_frmtp_case", "upper")` set to either 'upper' or 'lower' for the case of the 'P' for reporting p-values.
- `getOptions("qwraps2_frmtp_digits", 4)` Number of digits to the right of the decimal point to report p-values too. If `log10(p-value) < getOptions("qwraps2_frmtp_digits", 4)` then the output will be "P < 0.01", to however many digits are correct. Other options control other parts of the output p-value format.
- `getOptions("qwraps2_frmtp_leading0", TRUE)` to display or not to display the leading zero in p-values, i.e., if TRUE p-values are reported as 0.02 versus when FALSE p-values are reported as .02.
- `getOptions("qwraps2_journal", "default")` if a journal has specific formating for p-values or other statistics, this option will control the output. Many other options are ignored if this is any other than default. Check the github wiki, or this file, for current lists of implemented journal style methods.
- `getOptions("qwraps2_markup", latex)` value set to 'latex' or to 'markdowm'. Output is formatted to meet requirements of either markup language.
- `getOptions("qwraps2_style", "default")` By setting this option to a specific journal, p-values and other output, will be formatted to meet journal requirements.

Journals with predefined formatting

- Obstetrics \& Gynecology
  - http://www.editorialmanager.com/ong/default.aspx
  - `options(qwraps2_journal = "obstetrics_gynecology")`
  - P-value formating as of April 2015:
    Express P values to no more than three decimal places.
    Based on observations of published work, leading 0 will be omitted.
- Pediatric Dentistry:
  - http://www.aapd.org/publications/
  - `options(qwraps2_journal = "pediatric_dentistry")`
  - P-value formating as of March 2015.
    If P > .01, the actual value for P should be expressed to 2 digits. Non-significant values should not be expressed as "NS" whether or note P is significant, unless rounding a significant P-value expressed to 3 digits would make it non significant (ie P=.049, not P=.05). If P<.01, it should be express to 3 digits (eg, P=.003, not P<.05). Actual P-values should be expressed unless P<.001, in which case they should be so designated.

summary_table *Data Summary Tables*

Description

Tools useful for building data summary tables.

Usage

```
summary_table(.data, summaries)

tab_summary(x)

cbind.qwraps2_summary_table(..., deparse.level = 1)
```

Arguments

.data	a data.frame or grouped_df.
summaries	a list of lists of formulea for summarizing the data set.
x	a variable to summarize
...	qwraps2_summary_table objects to bind together
deparse.level	integer controlling the construction of labels in the case of non-matrix-like arguments (for the default method): deparse.level = 0 constructs no labels; the default, deparse.level = 1 or deparse.level = 2 constructs labels from the argument names.

Details

Detailed use of these functions can be found the a vignette.

The print method for the qwraps2_summary_table objects is just a simple wrapper for qable.

Value

a qwraps2_summary_table object.

See Also

qable for marking up qwraps2_data_summary objects. group_by for grouped_df objects.

cbind

Index