## Package ‘yarrr’

April 19, 2017

**Title**  A Companion to the e-Book "YaRrr!: The Pirate's Guide to R"

**Version**  0.1.5

**Date**  2017-4-18


**Depends**  jpeg, BayesFactor, circlize

**License**  GPL-2

**LazyData**  true

**URL**  www.thepiratesguidetor.com

**BugReports**  https://github.com/ndphillips/yarrr/issues

**RoxygenNote**  6.0.1

**Suggests**  knitr, rmarkdown

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Author**  Nathaniel Phillips [aut, cre]

**Maintainer**  Nathaniel Phillips <yarrr.book@gmail.com>

**Repository**  CRAN

**Date/Publication**  2017-04-19 08:39:31 UTC

### R topics documented:

- apa .................................................. 2
- auction .......................................... 3
- BeardLengths .................................... 3
- capture .......................................... 4
- diamonds ....................................... 5
- examscores .................................... 5
- movies .......................................... 6
- piratepal ........................................ 7
This function takes a hypothesis test object (e.g., \texttt{t.test()}, \texttt{cor.test()}, \texttt{chisq.test()}) as an input, and returns a string with the test result in APA format.

**Usage**

\[
\text{apa(}\text{test.object, tails} = 2, \text{sig.digits} = 2, \text{p.lb} = 0.01)\]

**Arguments**

- `test.object`: A hypothesis test object generated by functions such as \texttt{t.test()}, \texttt{cor.test()}, \texttt{chisq.test()}
- `tails`: The number of tails in the test (1 or 2)
- `sig.digits`: The number of digits results are rounded to
- `p.lb`: The lower bound of the p-value display. If the p-value is less than \texttt{p.lb}, the exact value will not be displayed.

**Examples**

\[
x \leftarrow \text{rnorm(}100\text{)}
y \leftarrow x + \text{rnorm(}100\text{)}
a \leftarrow \text{sample(}1:3, \text{size} = 200, \text{prob} = c(0.3, 0.2, 0.5), \text{replace} = \text{TRUE})
b \leftarrow \text{sample(}1:3, \text{size} = 200, \text{prob} = c(0.3, 0.2, 0.5), \text{replace} = \text{TRUE})
\]

\[
\text{apa(t.test}(x, y)\text{)}
\text{apa(cor.test}(x, y)\text{)}
\text{apa(chisq.test}(\text{table}(a, b))\text{)}
\]
**Description**

A dataframe containing data from 1000 ships sold at a pirate auction.

**Usage**

`auction`

**Format**

A data frame containing 1000 rows and 8 columns

- `cannons` (integer) The number of cannons on the ship
- `rooms` (integer) The number of rooms on the ship
- `age` (numeric) The age of the ship in years
- `condition` (integer) The condition of the ship on a scale of 1 to 10
- `color` (string) The color of the ship
- `style` (string) The style of the ship - either modern or classic
- `jbb` (numeric) The pre-sale predicted value of the ship according to Jack’s Blue Book (JBB)
- `price` (numeric) The actual selling price of the ship (in gold pieces, obviously)

**Source**

2015 annual pirate auction in Portland Oregon

---

**Description**

A dataframe containing the lengths of beards on 3 different pirate ships

**Usage**

`BeardLengths`

**Format**

A data frame containing 150 rows and 2 columns

- `Ship` (character) - The pirate’s ship
- `Beard` (numeric) - The length of the pirate’s beard in cm
Source

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany

---

**Description**

A dataframe containing a historical record of every ship the Perilous Pigeon captured on the Bodensee in the years 2014 and 2015

**Usage**

capture

**Format**

A data frame containing 1000 rows and 10 columns

- **size** (integer) - The size (length) of the ship (maybe in meters?)
- **cannons** (integer) - The number of cannons on the ship
- **style** (string) - The style of the ship (either modern or classic)
- **warnshot** (binary) - Did the ship fire a warning shot at the Perilous Pigeon when it got close?
- **date** (integer) - The date of the capture (1 = January 1, 365 = December 31)
- **heardof** (binary) - Was the target ship recognized by the captain’s first mate?
- **decorations** (integer) - An integer between 1 and 10 indicating how decorated the ship was. 1 means it looks totally common and shabby, 10 means it is among the finest looking ship you’ve ever seen!
- **daysfromshore** (integer) - How many days from the nearest land was the ship when it was found?
- **speed** (integer) - How fast was the ship going when it was caught?
- **treasure** (numeric) - How much treasure was found on the ship when it was captured?

**Source**

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany
### diamonds

<table>
<thead>
<tr>
<th>diamonds</th>
<th>diamonds</th>
</tr>
</thead>
</table>

**Description**
A dataframe containing information about 150 diamonds sold at auction.

**Usage**
diamonds

**Format**
A data frame containing 300 rows and 4 columns
- **weight** (numeric) - The weight of the diamond
- **clarity** (numeric) - The clarity of the diamond
- **color** (numeric) - The color shading of the diamond
- **value** The value of the diamond

**Source**
2015 annual international pirate meeting at the Bodensee in Konstanz, Germany

### examscores

<table>
<thead>
<tr>
<th>examscores</th>
<th>examscores</th>
</tr>
</thead>
</table>

**Description**
A dataframe containing the results of 4 exams given to 100 students. Each row represents a student, each column is a score on an exam.

**Usage**
examscores

**Format**
A data frame containing 100 rows and 4 columns
- **a** (numeric) - Score on exam a
- **b** (numeric) - ...exam b
- **c** (numeric) - ...exam c
- **d** (numeric) - ...exam d

**Source**
2015 annual international pirate meeting at the Bodensee in Konstanz, Germany
Description

A dataframe containing information about the top 5000 grossing movies of all time.

Usage

movies

Format

A data frame containing 5000 rows and 13 columns

name  Movie name
rating  MPAA rating
genre  Movie genre
creative.type  Creative type
time  Running time in minutes
year  Year of release
production.method  Production method
sequel  Was the movie a sequel? 1 = yes, 0 = no
budget  Movie budget (in $USD millions)
revenue.all  Gross worldwide revenue in $USD millions
revenue.dom  Domestic revenue in $USD millions
revenue.int  International revenue in $USD millions
revenue.inf  Inflation adjusted worldwide revenue in $USD millions

Source

www.the-numbers.com
piratepal

Description

This function provides a number of color palettes

Usage

piratepal(palette = "all", trans = 0, mix.col = "white", mix.p = 0,
          plot.result = FALSE, length.out = NULL)

Arguments

  palette  A string defining the color palette to use (see examples). To use a random
            palette, use "random". To plot all palettes, use "all". To see all palette names,
            use "names"
  trans    A number in the interval [0, 1] indicating how transparent to make the colors.
            A value of 0 means no transparency and a value of 1 means completely trans-
           parency.
  mix.col  string. An optional string representing a color to mix all colors in the palette with.
  mix.p    numeric. A number in the interval [0, 1] indicating how much to mix the palette colors with the color in mix.col
  plot.result A logical value indicating whether or not to display the colors.
  length.out An integer indicating how many colors to return. If length.out is larger than the number of colors in the palette, colors will be repeated.

Examples

  # Check out the vignette for a full guide
  vignette("piratepal", package = "yarrr")

  # Show all palettes
  piratepal(palette = "all")

  # Show some palettes
  piratepal(palette = "basel", trans = .5, plot.result = TRUE)

  # Using a palette in a scatterplot
nemo.cols <- piratepal(palette = "nemo", trans = .5)

set.seed(100) # For reproducibility
x <- rnorm(100)
y <- x + rnorm(100)

plot(x = x, y = y, col = nemo.cols,
     pch = 16,
     cex = runif(100, min = 0, max = 2),
     main = "piratepal('nemo', trans = .5)"
)

---

pirateplot

---

**Description**

The pirateplot function creates an RDI (Raw data, Descriptive and Inferential statistic) plot showing the relationship between 1 to 3 categorical independent variables and 1 continuous dependent variable.

**Usage**

```
pirateplot(formula = NULL, data = NULL, plot = TRUE,
           avg.line.fun = mean, pal = "basel", mix.col = "white", mix.p = 0,
           back.col = NULL, point.cex = NULL, point.pch = NULL, point.lwd = 1,
           jitter.val = 0.03, theme = 1, bean.b.o = NULL, bean.f.o = NULL,
           point.o = NULL, bar.f.o = NULL, bar.b.o = NULL, inf.f.o = NULL,
           inf.b.o = NULL, avg.line.o = NULL, gl.col = NULL, point.col = NULL,
           point.bg = NULL, bar.f.col = NULL, bean.b.col = NULL,
           bean.f.col = NULL, inf.f.col = NULL, inf.b.col = NULL,
           avg.line.col = NULL, bar.b.col = NULL, quant.col = NULL,
           avg.line.lwd = 4, bean.lwd = 1, bean.lty = 1, inf.lwd = NULL,
           bar.lwd = 1, at = NULL, bw = "nr0", adjust = 1, add = FALSE,
           sortx = "alphabetical", decreasing = FALSE, cex.lab = 1, cex.axis = 1,
           cex.names = 1, quant = NULL, quant.length = NULL, quant.lwd = NULL,
           quant.boxplot = FALSE, bty = "o", cap.beans = TRUE, family = NULL,
           inf.method = "hdi", inf.within = NULL, inf.p = NULL, hdi.iter = 1000,
           inf.disp = NULL, cut.min = NULL, cut.max = NULL, width.min = 0.3,
           width.max = NA, ylim = NULL, xlim = NULL, xlab = NULL, ylab = NULL,
           main = NULL, xaxt = NULL, xaxt = NULL, gl = NULL, gl.lwd = NULL,
           gl.lty = NULL, bar.b.lwd = NULL, line.fun = NULL, line.o = NULL,
           inf.o = NULL, bean.o = NULL, inf.col = NULL, theme.o = NULL,
           bar.o = NULL, inf = NULL, hdi.o = NULL, inf.type = NULL,
           inf.band = NULL)
```
Arguments

- **formula**
  A formula in the form $y \sim x1 + x2 + x3$ indicating the vertical response variable ($y$) and up to three independent variables.

- **data**
  Either a dataframe containing the variables specified in formula, a list of numeric vectors, or a numeric dataframe / matrix.

- **plot**
  Logical. If TRUE (the default), then the pirateplot is produced. If FALSE, the data summaries created in the plot are returned as a list.

- **avg.line.fun**
  Function. A function that determines how average lines and bar heights are determined (default is mean).

- **pal**
  String. The color palette of the plot. Can be a single color, a vector of colors, or the name of a palette in the piratepal() function (e.g., "basel", "google", "south-park"). To see all the palettes, run piratepal(palette = "all", action = "show")

- **mix.col, mix.p**
  Optional color mixing arguments to be passed to piratepal. See ?piratepal for examples.

- **back.col**
  String. Color of the plotting background.

- **point.cex, point.pch, point.lwd**
  Numeric. The size, pch type, and line width of raw data points.

- **jitter.val**
  Numeric. Amount of jitter added to points horizontally. Defaults to 0.05.

- **theme**
  Integer. An integer in the set 0, 1, 2 specifying a theme (that is, new default values for opacities and colors). theme = 0 turns off all opacities which can then be individually specified individually.

- **bar.f.o, point.o, inf.f.o, inf.b.o, avg.line.o, bean.b.o, bean.f.o, bar.b.o**
  Numeric. A number between 0 and 1 indicating how opaque to make the bars, points, inference band, average line, and beans respectively. These values override whatever is in the specified theme

- **point.col, bar.f.col, bean.b.col, bean.f.col, inf.f.col, inf.b.col, avg.line.col, bar.b.col, quant.col**
  String. Vectors of colors specifying the colors of the plotting elements. This will override values in the palette. f stands for filling, b stands for border.

- **bean.lwd, bean.lty, inf.lwd, avg.line.lwd, bar.lwd**
  Numeric. Vectors of numbers customizing the look of beans and lines.

- **at**
  Integer. Locations of the beans. Especially helpful when adding beans to an existing plot with add = TRUE

- **bw, adjust**
  Arguments passed to density calculations for beans (see ?density)

- **add**
  Logical. Should the pirateplot elements be added to an existing plotting space?

- **sortx**
  String. How to sort the x values. Can be "sequential" (as they are found in the original dataframe), "alphabetical", or a string in the set ("mean", "median", "min", "max") indicating a function

- **decreasing**
  Logical. If sortx is a named function, should values be sorted in decreasing order?

- **cex.lab, cex.axis, cex.names**
  Size of the labels, axes, and bean names.

- **quant**
  Numeric. Adds horizontal lines representing custom quantiles.

- **quant.length, quant.lwd**
  Numeric. Specifies line lengths/widths of quant.
quant.boxplot logical. Should standard values be included?

bty, xlim, ylim, xlab, ylab, main, yaxt, xaxt
General plotting arguments

cap.beans logical. Should maximum and minimum values of the bean densities be capped at the limits found in the data? Default is FALSE.

family a font family (Not currently in use)

inf.method string. A string indicating what types of inference bands to calculate. "ci" means frequentist confidence intervals, "hdi" means Bayesian Highest Density Intervals (HDI), "iqr" means interquartile range, "sd" means standard deviation, "se" means standard error, "withinci" means frequentist confidence intervals in a within design (Morey, 2008).

inf.within string. The variable which serves as an ID variable in a within design.

inf.p numeric. A number adjusting how inference ranges are calculated. For "ci" and "hdi", a number between 0 and 1 indicating the level of confidence (default is .95). For "sd" and "se", the number of standard deviations / standard errors added to or subtracted from the mean (default is 1).

hd.iter integer. Number of iterations to run when calculating the HDI. Larger values lead to better estimates, but can be more time consuming.

inf.disp string. How should inference ranges be displayed? "line" creates a classic vertical line, "rect" creates a rectangle, "bean" forms the inference around the bean.

cut.min, cut.max numeric. Optional minimum and maximum values of the beans.

width.min, width.max numeric. The minimum/maximum width of the beans.

gl numeric. Locations of the horizontal grid lines

bar.b.lwd, line.fun, inf.o, bean.o, inf.col, theme.o, inf, inf.type, inf.band, bar.o, line.o, hdi.o deprecated arguments

Examples

# Default pirateplot of weight by Time
pirateplot(formula = weight ~ Time,
    data = ChickWeight)

# Same but in grayscale
pirateplot(formula = weight ~ Time,
    data = ChickWeight,
    pal = "gray")
pirateplot

# Now using theme 2
pirateplot(formula = weight ~ time, 
data = ChickWeight, 
main = "Chicken weight by time", 
theme = 2) # theme 2

# theme 3
pirateplot(formula = weight ~ time, 
data = ChickWeight, 
main = "Chicken weight by time", 
theme = 3) # theme 3

# theme 4
pirateplot(formula = weight ~ time, 
data = ChickWeight, 
main = "Chicken weight by time", 
theme = 4) # theme 4

# Start with theme 2, but then customise!
pirateplot(formula = weight ~ time, 
data = ChickWeight, 
theme = 2, # theme 2 
pal = "xmen", # xmen palette 
main = "Chicken weights by Time", 
point.o = .4, # Add points 
point.col = "black", 
point.bg = "white", 
point.pch = 21, 
bean.f.o = .2, # Turn down bean filling 
inf.f.o = .8, # Turn up inf filling 
gl.col = "gray", # gridlines 
gl.lwd = c(.5, 0)) # turn off minor grid lines

# 2 IVs
pirateplot(formula = len ~ dose + supp, 
data = ToothGrowth, 
main = "Guinea pig tooth length by supplement", 
point.pch = 16, # Point specifications... 
point.col = "black", 
point.o = .7, 
inf.f.o = .9, # inference band opacity 
gl.col = "gray")

# Build everything from scratch with theme 0
# And use 3 IVs
pirateplot(formula = height ~ headband + eyepatch + sex, 
data = pirates, 
pal = gray(.1), # Dark gray palette 
theme = 0, # Start from scratch 
inf.f.o = .7, # Band opacity 
inf.f.col = piratepal("basel"), # Add color to bands 
point.o = .1, # Point opacity
avg.line.o = .8, # Average line opacity
gl.col = gray(.6), # Gridline specifications
gl.lty = 1,
gl.lwd = c(.5, 0))

# See the vignette for more details
vignette("pirateplot", package = "yarrr")

---

### Description

A dataset containing the results of a survey of 1,000 pirates.

### Usage

```r
pirates
```

### Format

A data frame containing 1,000 rows and 14 columns

- **id**: An integer giving the pirate’s id number
- **sex**: A string with the pirate’s self reported sex
- **age**: An integer giving the age of the pirate in years
- **height**: Height in cm
- **weight**: Weight in kg
- **headband**: A binary variable indicating whether the pirate wears a headband
- **college**: A string indicating the college the pirate went to. JSSFP stands for Jack Sparro’s School of Fashion and Piratery, while CCCC stands for Captain Chunk’s Cannon Crew
- **tattoos**: An integer indicating the number of tattoos the pirate has
- **tchests**: An integer indicating the number of treasure chests found by the pirate
- **parrots**: An integer indicating the number of parrots owned by the pirate in his/her lifetime
- **favorite.pirate**: A string indicating The pirate’s favorite pirate
- **sword.type**: A string indicating the type of sword the pirate uses
- **eyepatch**: An integer indicating the number of eyepatches worn by the pirate
- **sword.time**: A number indicating how long it takes (in seconds) for the pirate to draw his/her sword. Smaller times are better!
- **beard.length**: A number indicating length of the pirate’s beard in centimeters
- **fav.pixar**: A string indicating Pirate’s favorite pixar movie
- **grogg**: How many mugs of grogg the pirate drinks a day on average.
Source

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany

Description

A dataset containing the results of a survey of 1,000 pirates. This dataset is identical to the pirates dataset - except that it has many errors!

Usage

pirateserrors

Format

A data frame containing 1,000 rows and 14 columns

- **id**: An integer giving the pirate’s id number
- **sex**: A string with the pirate’s self reported sex
- **headband**: A binary variable indicating whether the pirate wears a headband
- **age**: An integer giving the age of the pirate in years
- **college**: A string indicating the college the pirate went to. JSSFP stands for Jack Sparro’s School of Fashion and Piratery, while CCCC stands for Captain Chunk’s Cannon Crew
- **tattoos**: An integer indicating the number of tattoos the pirate has
- **tchests**: An integer indicating the number of treasure chests found by the pirate
- **parrots**: An integer indicating the number of parrots owned by the pirate in his/her lifetime
- **favorite.pirate**: A string indicating the pirate’s favorite pirate
- **sword.type**: A string indicating the type of sword the pirate uses
- **sword.time**: A number indicating how long it takes (in seconds) for the pirate to draw his/her sword. Smaller times are better!
- **eyepatch**: An integer indicating the number of eyepatches worn by the pirate
- **beard.length**: A number indicating length of the pirate’s beard in centimeters
- **fav.pixar**: A string indicating the pirate’s favorite pixar movie

Source

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany
**Description**

A dataframe containing travel times of chartered ships from three pirate companies to three different destinations.

**Usage**

`pircharter`

**Format**

A data frame containing 1000 rows and 10 columns

- **company** (string) - The charter company: JoRo = Jolly Roger, BmcB = Boaty McBoat, MiPa = Millenium Parrot
- **destination** (string) - The destination of the charter
- **time** (numeric) - The travel time of the ship in hours

**Source**

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany

**Description**

A dataframe containing the amount of time it took to clean both pirate and shark poop from the poop deck using one of three different cleaning solutions

**Usage**

`poopdeck`

**Format**

A data frame containing 300 rows and 4 columns

- **day** (factor) - The day that the poop deck was cleaned (1 through 10000)
- **cleaner** (string) - The cleaning solution used
- **type** (string) - The type of poop being cleaned
- **time** (numeric) - The amount of time (in minutes) the cleaning took.
Source

2015 annual international pirate meeting at the Bodensee in Konstanz, Germany

---

**recodev**

**Description**

This function takes a vector original.vector, and converts all values in a vector old.values to the values in a new vector new.values.

**Usage**

recodev(original.vector, old.values, new.values, others = NULL)

**Arguments**

- original.vector: A vector you want to recode
- old.values: A vector of length M.
- new.values: A vector of length M.
- others: An optional value indicating what to convert all values in original.vector that are not found in old.values.

**Examples**

```r
x <- c("y", "y", "XSF", "y", "0", "X", "0", "y", "n", "0", "1", "1")
recodev(original.vector = x,
        old.values = c("y", "1", "n", "0"),
        new.values = c(1, 1, 0, 0)
)
```

```r
x <- c("y", "y", "XSF", "y", "0", "X", "0", "y", "n", "0", "1", "1")
recodev(original.vector = x,
        old.values = c("y", "1", "n", "0"),
        new.values = c(1, 1, 0, 0),
        others = NA
)
```
transparent function

Description

This function takes a standard color as an argument and returns a transparent version of that color.

Usage

transparent(orig.col = "red", trans.val = 1, maxColorValue = 255)

Arguments

- **orig.col**: The original color to be made transparent. Can be specified as a string or a vector of rgb values.
- **trans.val**: A number in the interval [0, 1] indicating how transparent to make the color.
- **maxColorValue**: The maximum color value (only used when orig.col is an rgb vector).

Examples

```r
# Diagram of some examples
plot(1, ylim = c(0, 1), xlim = c(0, 12), bty = "n",
     xaxt = "n", yaxt = "n", ylab = "", xlab = "", type = "n")

text(6, .9, "transparent('red', trans.val = x)")
points(x = 1:11, y = rep(.8, 11), pch = 16,
       col = transparent("red", seq(0, 1, .1)), cex = 2)

text(6, .7, "transparent('red', trans.val = x)")
points(x = 1:11, y = rep(.6, 11), pch = 16,
       col = transparent("blue", seq(0, 1, .1)), cex = 2)

# Scatterplot with transparent colors
```
yarrr.guide

   a.x <- rnorm(100, mean = 0, sd = 1)
   a.y <- a.x + rnorm(100, mean = 0, sd = 1)

   par(mfrow = c(3, 3))

   for(trans.val.i in seq(0, .1, length.out = 9)) {

      plot(a.x, a.y, pch = 16, col = transparent("blue", trans.val.i), cex = 1.5,
           xlim = c(-5, 5), ylim = c(-5, 5), xlab = "x", ylab = "y",
           main = paste("trans.val = ", round(trans.val.i, 2), sep = ""))
   }

---

yarrr.guide | Opens the HTML manual for the yarrr package

**Description**

Opens the HTML manual for the yarrr package

**Usage**

yarrr.guide()
Index

*Topic **apa**
apa, 2

*Topic **colors**
piratepal, 7
transparent, 16

*Topic **datasets**
auction, 3
BeardLengths, 3
capture, 4
diamonds, 5
examscores, 5
movies, 6
pirates, 12
pirateserrors, 13
pircharter, 14
poopdeck, 14

*Topic **misc**
yarrr.guide, 17

*Topic **plot**
pirateplot, 8

apa, 2
auction, 3
BeardLengths, 3
capture, 4
diamonds, 5
examscores, 5
movies, 6
piratepal, 7
pirateplot, 8
pirates, 12
pirateserrors, 13
pircharter, 14
poopdeck, 14
recode, 15
transparent, 16
yarrr.guide, 17