Package 'bayes4psy'

July 22, 2025

Version 1.2.13

Title User Friendly Bayesian Data Analysis for Psychology

Description Contains several Bayesian models for data analysis of psychological tests. A user friendly interface for these models should enable students and researchers to perform professional level Bayesian data analysis without advanced knowledge in programming and Bayesian statistics. This package is based on the Stan platform (Carpenter et el. 2017 <doi:10.18637/jss.v076.i01>).

License GPL (>= 3)

Encoding UTF-8

LazyData true

ByteCompile true

Depends methods (>= 4.0.0), R (>= 4.0.0), Rcpp (>= 1.0.5)

Imports circular (>= 0.4.93), cowplot (>= 1.1.0), dplyr (>= 1.0.2), emg (>= 1.0.9), ggplot2 (>= 3.3.2), metRology (>= 0.9.28.1), reshape (>= 0.8.8), rstan (>= 2.26.0), rstantools (>= 2.1.1), mcmcse (>= 1.4.1), stats (>= 4.0.0), RcppParallel

Suggests testthat (>= 3.0.0), rmarkdown (>= 2.5.0), knitr (>= 1.30.0)

LinkingTo StanHeaders (>= 2.26.0), rstan (>= 2.26.0), BH (>= 1.72.0.3), Rcpp (>= 1.0.5), RcppEigen (>= 0.3.3.7.0), RcppParallel

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

UseLTO true

RoxygenNote 7.2.3

URL https://github.com/bstatcomp/bayes4psy

BugReports https://github.com/bstatcomp/bayes4psy/issues

Author Jure Demšar [cre, aut],
Grega Repovš [aut],
Erik Štrumbelj [aut],

Trustees of Columbia University [cph],
John Kruschke [cph] (R/shared_functions.R - mcmc_hdi,
src/stan_files/ttest.stan),
Rasmus Baath [cph] (R/b_bootstrap.R)

Maintainer Jure Demšar <jure.demsar@fri.uni-lj.si>

Repository CRAN

Date/Publication 2025-07-16 10:30:02 UTC

Contents

`bayes4psy-package` *The 'bayes4psy' package.*

Description

A user-friendly implementation of Bayesian statistical methods commonly used in social sciences. All used models are pre-compiled, meaning that users only need to call appropriate functions using their data.

References

Stan Development Team (NA) - the Stan framework and RStan interface. John Kruschke - mcmc_hdi function Rasmus Bååth - Easy Bayesian Bootstrap in R

`bayes4psy-datasets` *Datasets for bayes4psy examples Example datasets for use in* **rstanarm** *examples and vignettes. The datasets were extracted from the internal MBLab* `http://www.mblab.si` *repository. MBLab is a research lab at the Faculty of Arts, Department of Psychology, University of Ljubljana, Slovenia.*

Description

Datasets for bayes4psy examples Example datasets for use in **rstanarm** examples and vignettes. The datasets were extracted from the internal MBLab `http://www.mblab.si` repository. MBLab is a research lab at the Faculty of Arts, Department of Psychology, University of Ljubljana, Slovenia.

Format

`adaptation_level_small` Small dataset on subjects picking up weights and determining their weights from 1..10.

Source: Internal MBLab repository.

50 obs. of 3 variables

- `sequence` sequence index.

- weight actual weight of the object.
- response subject's estimation of weight.

adaptation_level Data on subjects picking up weights and determining their weights from 1..10.

Source: Internal MBLab repository.

2900 obs. of 6 variables

- subject subject index.
- group group index.
- part first or second part of the experiment.
- sequence sequence index.
- weight actual weight of the object.
- response subject's estimation of weight.

#'

after_images_opponent_process Colors predicted by the opponent process theory.

Source: Internal MBLab repository.

6 obs. of 7 variables

- stimuli name of the color stimuli.
- r value of the R component in the RGB model.
- g value of the G component in the RGB model.
- b value of the B component in the RGB model.
- h value of the H component in the HSV model.
- s value of the S component in the HSV model.
- v value of the V component in the HSV model.

#'

after_images_opponent_stimuli Stimuli used in the after images experiment.

Source: Internal MBLab repository.

6 obs. of 7 variables

- r_s value of the R component in the RGB model.
- g_s value of the G component in the RGB model.
- b_s value of the B component in the RGB model.
- stimuli name of the color stimuli.
- h_s value of the H component in the HSV model.
- s_s value of the S component in the HSV model.
- v_s value of the V component in the HSV model.

#'

after_images_trichromatic Colors predicted by the trichromatic theory.

Source: Internal MBLab repository.

6 obs. of 7 variables

- stimuli name of the color stimuli.
- r value of the R component in the RGB model.
- g value of the G component in the RGB model.
- b value of the B component in the RGB model.

- h value of the H component in the HSV model.
- s value of the S component in the HSV model.
- v value of the V component in the HSV model.

#'

after_images Data gathered by the after images experiment.

Source: Internal MBLab repository.

1311 obs. of 12 variables

- subject subject index.
- rt reaction time.
- r value of the R component in the RGB model of subject's response.
- g value of the G component in the RGB model of subject's response.
- b value of the B component in the RGB model of subject's response.
- stimuli name of the color stimuli.
- r_s value of the R component in the RGB model of the shown stimulus
- g_s value of the G component in the RGB model of the shown stimulus
- b_s value of the B component in the RGB model of the shown stimulus
- h_s value of the H component in the HSV model of the shown stimulus
- s_s value of the S component in the HSV model of the shown stimulus
- v_s value of the V component in the HSV model of the shown stimulus

#'

flanker Data gathered by the flanker experiment.

Source: Internal MBLab repository.

8256 obs. of 5 variables

- subject subject index.
- group group index.
- congruencty type of stimulus.
- result was subject's reponse correct or wrong?
- rt reaction time.

#'

stroop_extended All the data gathered by the Stroop experiment.

Source: Internal MBLab repository.

41068 obs. of 5 variables

- subject subject ID.
- cond type of condition.
- rt reaction time.
- acc was subject's reponse correct or wrong?
- age age of subject.

#'

stroop_simple All the data gathered by the Stroop experiment.

Source: Internal MBLab repository.

61 obs. of 5 variables

- `subject` subject ID.
- `reading_neutral` average response time for reading neutral stimuli.
- `naming_neutral` average response time for naming neutral stimuli.
- `reading_incongruent` average response time for reading incongruent stimuli.
- `naming_incongruent` average response time for naming incongruent stimuli.

Examples

```
# Example of Bayesian bootstraping on 'adaptation_level_small' dataset
# linear function of seqence vs. response
lm_statistic <- function(data) {
  lm(sequence ~ response, data)$coef
}

# load data
data <- adaptation_level_small

# bootstrap
data_bootstrap <- b_bootstrap(data, lm_statistic, n1=1000, n2=1000)
```

`b_bootstrap` *b_bootstrap*

Description

Performs a Bayesian bootstrap and returns a sample of size n1 representing the posterior distribution of the statistic. Returns a vector if the statistic is one-dimensional (like for mean(...)) or a data.frame if the statistic is multi-dimensional (like for the coefficients of lm).

Usage

```
b_bootstrap(
  data,
  statistic,
  n1 = 1000,
  n2 = 1000,
  use_weights = FALSE,
  weight_arg = NULL,
  ...
)
```

Arguments

`data`	The data as either a vector, matrix or data.frame.
`statistic`	A function that accepts data as its first argument and if use_weights is TRUE the weights as its second argument. Function should return a numeric vector.

n1	The size of the bootstrap sample (default = 1000).
n2	The sample size used to calculate the statistic each bootstrap draw (default = 1000).
use_weights	Whether the statistic function accepts a weight argument or should be calculated using resampled data (default = FALSE).
weight_arg	If the statistic function includes a named argument for the weights this could be specified here (default = NULL).
...	Further arguments passed on to the statistic function.

Value

A data frame containing bootstrap samples.

Author(s)

Rasmus Baath

References

https://www.sumsar.net/blog/2015/07/easy-bayesian-bootstrap-in-r/

Rubin, D. B. (1981). The Bayesian Bootstrap. The annals of statistics, 9(1), 130-134.

Examples

```
# linear function of seqence vs. response
lm_statistic <- function(data) {
  lm(sequence ~ response, data)$coef
}

# load data
data <- adaptation_level_small

# bootstrap
data_bootstrap <- b_bootstrap(data, lm_statistic, n1 = 1000, n2 = 1000)
```

---

b_color *b_color*

Description

Bayesian model for comparing colors.

Usage

```
b_color(
  colors,
  priors = NULL,
  hsv = FALSE,
  warmup = 1000,
  iter = 2000,
  chains = 4,
  seed = NULL,
  refresh = NULL,
  control = NULL,
  suppress_warnings = TRUE
)
```

Arguments

colors	a data frame of colors either in RGB or HSV format. The first column should be the R (or H) component, the second column should be the G (or S) component and the third column should be the B (or V) component.
priors	List of parameters and their priors - b_prior objects. You can put a prior on the mu_r (mean r component), sigma_r (variance of mu_r), mu_g (mean g component), sigma_g (variance of mu_g), mu_b (mean b component), sigma_b (variance of mu_b), mu_h (mean h component), kappa_h (variance of mu_h), mu_s (mean s component), sigma_s (variance of mu_s), mu_v (mean v component) and sigma_v (variance of mu_v) parameters (default = NULL).
hsv	set to TRUE if colors are provided in HSV format (default = FALSE).
warmup	Integer specifying the number of warmup iterations per chain (default = 1000).
iter	Integer specifying the number of iterations (including warmup, default = 2000).
chains	Integer specifying the number of parallel chains (default = 4).
seed	Random number generator seed (default = NULL).
refresh	Frequency of output (default = NULL).
control	A named list of parameters to control the sampler's behavior (default = NULL).
suppress_warnings	Suppress warnings returned by Stan (default = TRUE).

Value

An object of class 'color_class'

Examples

```
# priors for rgb
mu_prior <- b_prior(family="uniform", pars=c(0, 255))
sigma_prior <- b_prior(family="uniform", pars=c(0, 100))

# attach priors to relevant parameters
```

```
priors_rgb <- list(c("mu_r", mu_prior),
                   c("sigma_r", sigma_prior),
                   c("mu_g", mu_prior),
                   c("sigma_g", sigma_prior),
                   c("mu_b", mu_prior),
                   c("sigma_b", sigma_prior))


# generate data (rgb)
r <- as.integer(rnorm(100, mean=250, sd=20))
r[r > 255] <- 255
r[r < 0] <- 0

g <- as.integer(rnorm(100, mean=20, sd=20))
g[g > 255] <- 255
g[g < 0] <- 0

b <- as.integer(rnorm(100, mean=40, sd=20))
b[b > 255] <- 255
b[b < 0] <- 0

colors_rgb <- data.frame(r=r, g=g, b=b)

# fit
fit_rgb <- b_color(colors=colors_rgb, priors=priors_rgb, chains=1)


# priors for hsv
h_prior <- b_prior(family="uniform", pars=c(0, 2*pi))
sv_prior <- b_prior(family="uniform", pars=c(0, 1))
kappa_prior <- b_prior(family="uniform", pars=c(0, 500))
sigma_prior <- b_prior(family="uniform", pars=c(0, 1))

# attach priors to relevant parameters
priors_hsv <- list(c("mu_h", h_prior),
                   c("kappa_h", kappa_prior),
                   c("mu_s", sv_prior),
                   c("sigma_s", sigma_prior),
                   c("mu_v", sv_prior),
                   c("sigma_v", sigma_prior))

# generate data (hsv)
h <- rnorm(100, mean=2*pi/3, sd=0.5)
h[h > 2*pi] <- 2*pi
h[h < 0] <- 0

s <- rnorm(100, mean=0.9, sd=0.2)
s[s > 1] <- 1
s[s < 0] <- 0

v <- rnorm(100, mean=0.9, sd=0.2)
v[v > 1] <- 1
v[v < 0] <- 0
```

```
colors_hsv <- data.frame(h=h, s=s, v=v)

# fit
fit_hsv <- b_color(colors=colors_hsv, hsv=TRUE, priors=priors_hsv, chains=1)
```

---

b_linear *b_linear*

Description

Bayesian model for fitting a linear normal model to data.

Usage

```
b_linear(
  x,
  y,
  s,
  priors = NULL,
  warmup = 1000,
  iter = 2000,
  chains = 4,
  seed = NULL,
  refresh = NULL,
  control = NULL,
  suppress_warnings = TRUE
)
```

Arguments

x	a vector containing sequence indexes (time).
y	a vector containing responses of subjects.
s	a vector containing subject indexes. Starting index should be 1 and the largest subject index should equal the number of subjects.
priors	List of parameters and their priors - b_prior objects. You can put a prior on the mu_a (mean intercept), sigma_a (variance of mu_a), mu_b (mean slope), sigma_s (variance of mu_b), mu_s (variance) and sigma_s (variance of mu_s) parameters (default = NULL).
warmup	Integer specifying the number of warmup iterations per chain (default = 1000).
iter	Integer specifying the number of iterations (including warmup, default = 2000).
chains	Integer specifying the number of parallel chains (default = 4).
seed	Random number generator seed (default = NULL).

refresh Frequency of output (default = NULL).

control A named list of parameters to control the sampler's behavior (default = NULL).

suppress_warnings
Suppress warnings returned by Stan (default = TRUE).

Value

An object of class 'linear_class'.

Examples

```
# priors
mu_prior <- b_prior(family="normal", pars=c(0, 100))
sigma_prior <- b_prior(family="uniform", pars=c(0, 500))

# attach priors to relevant parameters
priors <- list(c("mu_a", mu_prior),
               c("sigma_a", sigma_prior),
               c("mu_b", mu_prior),
               c("sigma_b", sigma_prior),
               c("mu_s", sigma_prior),
               c("sigma_s", sigma_prior))

# generate data
x <- vector()
y <- vector()
s <- vector()
for (i in 1:5) {
  x <- c(x, rep(1:10, 2))
  y <- c(y, rnorm(20, mean=1:10, sd=2))
  s <- c(s, rep(i, 20))
}

fit <- b_linear(x=x, y=y, s=s, priors=priors, chains=1)
```

---

b_prior-class *b_prior*

Description

An S4 class for defining priors for Bayesian models.

Slots

family Prior family - \"uniform\", \"normal\", \"gamma\" or \"beta\".

pars Parameters of the prior - a vector of two numerical values.

b_reaction_time *b_reaction_time*

Description

Bayesian model for comparing reaction times.

Usage

```
b_reaction_time(
  t,
  s,
  priors = NULL,
  warmup = 1000,
  iter = 2000,
  chains = 4,
  seed = NULL,
  refresh = NULL,
  control = NULL,
  suppress_warnings = TRUE
)
```

Arguments

t	a vector containing reaction times for each measurement.
s	a vector containing subject indexes. Starting index should be 1 and the largest subject index should equal the number of subjects.
priors	List of parameters and their priors - b_prior objects. You can put a prior on the mu_m (mean), sigma_m (variance of mu_m), mu_s (variance), sigma_s (variance of mu_s), mu_l (mean of the exponent factor) and sigma_l (variance of mu_l) parameters (default = NULL).
warmup	Integer specifying the number of warmup iterations per chain (default = 1000).
iter	Integer specifying the number of iterations (including warmup, default = 2000).
chains	Integer specifying the number of parallel chains (default = 4).
seed	Random number generator seed (default = NULL).
refresh	Frequency of output (default = NULL).
control	A named list of parameters to control the sampler's behavior (default = NULL).
suppress_warnings	Suppress warnings returned by Stan (default = TRUE).

Value

An object of class 'reaction_time_class'

Examples

```
# priors
mu_prior <- b_prior(family="normal", pars=c(0, 100))
sigma_prior <- b_prior(family="uniform", pars=c(0, 500))
lambda_prior <- b_prior(family="uniform", pars=c(0.05, 5))

# attach priors to relevant parameters
priors <- list(c("mu_m", mu_prior),
               c("sigma_m", sigma_prior),
               c("mu_s", sigma_prior),
               c("sigma_s", sigma_prior),
               c("mu_l", lambda_prior),
               c("sigma_l", sigma_prior))

# generate data
s <- rep(1:5, 20)
rt <- emg::remg(100, mu=10, sigma=1, lambda=0.4)

# fit
fit <- b_reaction_time(t=rt, s=s, priors=priors, chains=1)
```

---

b_results-class *b_results*

Description

Parent S4 class for declaring shared function generics.

---

b_success_rate *b_success_rate*

Description

Bayesian model for comparing test success rate.

Usage

```
b_success_rate(
  r,
  s,
  priors = NULL,
  warmup = 1000,
  iter = 2000,
  chains = 4,
```

```
  seed = NULL,
  refresh = NULL,
  control = NULL,
  suppress_warnings = TRUE
)
```

Arguments

r	a vector containing test results (0 - test was not solved successfully, 1 - test was solved successfully).
s	a vector containing subject indexes. Starting index should be 1 and the largest subject index should equal the number of subjects.
priors	List of parameters and their priors - b_prior objects. You can put a prior on the p (mean probability of success) and tau (variance) parameters (default = NULL).
warmup	Integer specifying the number of warmup iterations per chain (default = 1000).
iter	Integer specifying the number of iterations (including warmup, default = 2000).
chains	Integer specifying the number of parallel chains (default = 4).
seed	Random number generator seed (default = NULL).
refresh	Frequency of output (default = NULL).
control	A named list of parameters to control the sampler's behavior (default = NULL).
suppress_warnings	Suppress warnings returned by Stan (default = TRUE).

Value

An object of class 'success_rate_class'.

Examples

```
# priors
p_prior <- b_prior(family="beta", pars=c(1, 1))
tau_prior <- b_prior(family="uniform", pars=c(0, 500))

# attach priors to relevant parameters
priors <- list(c("p", p_prior),
               c("tau", tau_prior))

# generate data
s <- rep(1:5, 20)
data <- rbinom(100, size=1, prob=0.6)

# fit
fit <- b_success_rate(r=data, s=s, priors=priors, chains=1)
```

b_ttest *b_ttest*

Description

Bayesian t-test.

Usage

```
b_ttest(
  data,
  priors = NULL,
  warmup = 1000,
  iter = 2000,
  chains = 4,
  seed = NULL,
  refresh = NULL,
  control = NULL,
  suppress_warnings = TRUE
)
```

Arguments

data	Numeric vector of values on which the fit will be based.
priors	List of parameters and their priors - b_prior objects. You can put a prior on the mu (mean) and sigma (variance) parameters (default = NULL).
warmup	Integer specifying the number of warmup iterations per chain (default = 1000).
iter	Integer specifying the number of iterations (including warmup, default = 2000).
chains	Integer specifying the number of parallel chains (default = 4).
seed	Random number generator seed (default = NULL).
refresh	Frequency of output (default = NULL).
control	A named list of parameters to control the sampler's behavior (default = NULL).
suppress_warnings	Suppress warnings returned by Stan (default = TRUE).

Value

An object of class 'ttest_class'.

Examples

```
# priors
mu_prior <- b_prior(family="normal", pars=c(0, 1000))
sigma_prior <- b_prior(family="uniform", pars=c(0, 500))
nu_prior <- b_prior(family="normal", pars=c(2000, 1000))
```

```
# attach priors to relevant parameters
priors <- list(c("mu", mu_prior),
               c("sigma", sigma_prior),
               c("nu", nu_prior))

# generate some data
data  <- rnorm(20, mean=150, sd=20)

# fit
fit <- b_ttest(data=data, priors=priors, chains=1)
```

---

color_class-class *color_class*

Description

An S4 class for storing results of Bayesian color model.

Functions

summary(‘color_class‘): prints a summary of the fit.

print(‘color_class‘): prints a more detailed summary of the fit

show(‘color_class‘): prints a more detailed summary of the fit.

plot(‘color_class‘): plots fitted model against the data. Use this function to explore the quality of your fit. You can compare fit with underlying data only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_fit(‘color_class‘): plots fitted model against the data. Use this function to explore the quality of your fit. You can compare fit with underlying data only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_trace(‘color_class‘): traceplot for main fitted model parameters.

get_parameters(‘color_class‘): returns a dataframe with values of fitted parameters.

compare_means(‘color_class‘, fit2=‘color_class‘): prints color difference between two fits. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

compare_means(‘color_class‘, rgb=‘vector‘): prints color difference between a fit and a color defined with rgb components. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

compare_means(‘color_class‘, hsv=‘vector‘): prints color difference between a fit and a color defined with hsv components. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means_difference(‘color_class‘, fit2=‘color_class‘): a visualization of the difference between two fits You can also provide the rope and bins (number of bins in the histogram) parameters or

visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means_difference(‘color_class‘, rgb=‘vector‘): a visualization of the difference between a fit and a color defined with rgb components. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means_difference(‘color_class‘, hsv=‘vector‘): a visualization of the difference between a fit and a color defined with hsv components. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means(‘color_class‘): plots density of means. You can also visualize the density only for chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means(‘color_class‘, fit2=‘color_class‘): plots density for the first and the second group means. You can also visualize the density only for chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means(‘color_class‘, rgb=‘vector‘): plots density for the first and a color defined with rgb components. You can also visualize the density only for chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_means(‘color_class‘, hsv=‘vector‘): plots density for the first and a color defined with hsv components. You can also visualize the density only for chosen color components (r, g, b, h, s, v) by using the pars parameter.

compare_distributions(‘color_class‘, fit2=‘color_class‘): draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

compare_distributions(‘color_class‘, rgb=‘vector‘): draws samples from distribution of the first group and compares them against a color defined with rgb components. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

compare_distributions(‘color_class‘, hsv=‘vector‘): draws samples from distribution of the first group and compares them against a color defined with hsv components. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions(‘color_class‘): a visualization of the fitted distribution. You can visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions(‘color_class‘, fit2=‘color_class‘): a visualization of two fitted distributions. You can visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions(‘color_class‘, rgb=‘vector‘): a visualization of the fitted distribution and a color defined with rgb components. You can visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions(‘color_class‘, hsv=‘vector‘): a visualization of the fitted distribution and a color defined with hsv components. You can visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions_difference(‘color_class‘, fit2=‘color_class‘): a visualization of the difference between the distribution of the first fit and the second fit. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions_difference(‘color_class‘, rgb=‘vector‘): a visualization of the difference between the distribution of the first fit and a color defined with rgb components. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_distributions_difference(‘color_class‘, hsv=‘vector‘): a visualization of the difference between the distribution of the first fit and a color defined with hsv components. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through chosen color components (r, g, b, h, s, v) by using the pars parameter.

plot_hsv(‘color_class‘): plots fitted model against the data. Use this function to explore the quality of your fit thorough a circular visualization of hsv color components.

plot_fit_hsv(‘color_class‘): plots fitted model against the data. Use this function to explore the quality of your fit thorough a circular visualization of hsv color components.

plot_means_hsv(‘color_class‘): a visualization of the difference between means of two fits through a circular visualization of hsv color components. You can also compare fit means with colors defined through rgb or hsv components (as points or as lines on the visualization).

plot_distributions_hsv(‘color_class‘): a visualization of distributions of one or two fits thorough a circular visualization of hsv color components. You can also compare fit means with colors defined through rgb or hsv components (as points or as lines on the visualization).

Slots

`extract` Extract from Stan fit.

`fit` Stan fit.

`data` Data on which the fit is based.

Examples

```
# priors for rgb
mu_prior <- b_prior(family="uniform", pars=c(0, 255))
sigma_prior <- b_prior(family="uniform", pars=c(0, 100))

# attach priors to relevant parameters
priors_rgb <- list(c("mu_r", mu_prior),
                   c("sigma_r", sigma_prior),
                   c("mu_g", mu_prior),
                   c("sigma_g", sigma_prior),
                   c("mu_b", mu_prior),
                   c("sigma_b", sigma_prior))


# generate data (rgb) and fit
r <- as.integer(rnorm(100, mean=250, sd=20))
r[r > 255] <- 255
r[r < 0] <- 0
```

```
g <- as.integer(rnorm(100, mean=20, sd=20))
g[g > 255] <- 255
g[g < 0] <- 0

b <- as.integer(rnorm(100, mean=40, sd=20))
b[b > 255] <- 255
b[b < 0] <- 0

colors <- data.frame(r=r, g=g, b=b)

fit1 <- b_color(colors=colors, priors=priors_rgb, chains=1)


# priors for hsv
h_prior <- b_prior(family="uniform", pars=c(0, 2*pi))
sv_prior <- b_prior(family="uniform", pars=c(0, 1))
kappa_prior <- b_prior(family="uniform", pars=c(0, 500))
sigma_prior <- b_prior(family="uniform", pars=c(0, 1))

# attach priors to relevant parameters
priors_hsv <- list(c("mu_h", h_prior),
                   c("kappa_h", kappa_prior),
                   c("mu_s", sv_prior),
                   c("sigma_s", sigma_prior),
                   c("mu_v", sv_prior),
                   c("sigma_v", sigma_prior))

# generate data (hsv) and fit
h <- rnorm(100, mean=2*pi/3, sd=0.5)
h[h > 2*pi] <- 2*pi
h[h < 0] <- 0

s <- rnorm(100, mean=0.9, sd=0.2)
s[s > 1] <- 1
s[s < 0] <- 0

v <- rnorm(100, mean=0.9, sd=0.2)
v[v > 1] <- 1
v[v < 0] <- 0

colors <- data.frame(h=h, s=s, v=v)

fit2 <- b_color(colors=colors, hsv=TRUE, priors=priors_hsv, chains=1)


# a short summary of fitted parameters
summary(fit1)

# a more detailed summary of fitted parameters
print(fit1)
show(fit1)
```

```
# plot the fitted distribution against the data
plot(fit1)
plot_fit(fit1)

# plot the fitted distribution against the data,
# specify only a subset of parameters for visualization
plot(fit1, pars=c("h", "s", "v"))
plot_fit(fit1, pars=c("h", "s", "v"))

# traceplot of the fitted parameters
plot_trace(fit1)

# extract parameter values from the fit
parameters <- get_parameters(fit1)

# compare means between two fits
compare_means(fit1, fit2=fit2)

# compare means between two fits,
# specify only a subset of parameters for comparison
compare_means(fit1, fit2=fit2, pars=c("h", "s", "v"))

# compare means of a fit with an rgb defined color
compare_means(fit1, rgb=c(255, 0, 0))

# compare means of a fit with an hsv defined color
compare_means(fit1, hsv=c(pi/2, 1, 1))

# visualize difference in means between two fits
plot_means_difference(fit1, fit2=fit2)

# visualize difference in means between two fits,
# specify only a subset of parameters for comparison, use a rope interval
plot_means_difference(fit1, fit2=fit2, pars=c("r", "g", "b"), rope=10)

# visualize difference in means between a fit and an rgb defined color
plot_means_difference(fit1, rgb=c(255, 0, 0))

# visualize difference in means between a fit and an hsv defined color
plot_means_difference(fit1, hsv=c(pi/2, 1, 1))

# visualize means of a single fit
plot_means(fit1)

# visualize means of two fits
plot_means(fit1, fit2=fit2)

# visualize means of two fits,
# specify only a subset of parameters for visualization
plot_means(fit1, fit2=fit2, pars=c("h", "s", "v"))

# visualize means of a single fit and an rgb defined color
plot_means(fit1, rgb=c(255, 0, 0))
```

```
# visualize means of a single fit and an an hsv defined color
plot_means(fit1, hsv=c(pi/2, 1, 1))

# draw samples from distributions underlying two fits and compare them
compare_distributions(fit1, fit2=fit2)

# draw samples from distributions underlying two fits and compare them,
# specify only a subset of parameters for comparison, use a rope interval
compare_distributions(fit1, fit2=fit2, pars=c("r", "g", "b"), rope=10)

# draw samples from a distribution underlying the fits,
# compare them with an rgb defined color
compare_distributions(fit1, rgb=c(255, 0, 0))

# draw samples from a distribution underlying the fits,
# compare them with an hsv defined color
compare_distributions(fit1, hsv=c(pi/2, 1, 1))

# visualize the distribution underlying a fit
plot_distributions(fit1)

# visualize distributions underlying two fits
plot_distributions(fit1, fit2=fit2)

# visualize distributions underlying two fits,
# specify only a subset of parameters for visualization
plot_distributions(fit1, fit2=fit2, pars=c("h", "s", "v"))

# visualize the distribution underlying a fit, and an rgb defined color
plot_distributions(fit1, rgb=c(255, 0, 0))

# visualize the distribution underlying a fit, and an hsv defined color
plot_distributions(fit1, hsv=c(pi/2, 1, 1))

# visualize difference between distributions underlying two fits
plot_distributions_difference(fit1, fit2=fit2)

# visualize difference between distributions underlying two fits
# specify only a subset of parameters for comparison, use a rope interval
plot_distributions_difference(fit1, fit2=fit2, pars=c("r", "g", "b"), rope=10)

# visualize difference between the distributions underlyin a fit,
# and an rgb defined color
plot_distributions_difference(fit1, rgb=c(255, 0, 0))

# visualize difference between the distributions underlyin a fit,
# and an hsv defined color
plot_distributions_difference(fit1, hsv=c(pi/2, 1, 1))

# plot the fitted distribution for hue against the hue data
plot_hsv(fit1)
```

```
# plot the fitted distribution for hue against the hue data
plot_fit_hsv(fit1)

# visualize hue means of a single fit
plot_means_hsv(fit1)

# visualize hue means of two fits
plot_means_hsv(fit1, fit2=fit2)

# visualize hue means of two fits, add annotation points and lines,
# hsv parameter determines whether annotations are defined in hsv or rgb
lines <- list()
lines[[1]] <- c(2*pi, 1, 1)
lines[[2]] <- c(pi/2, 0.5, 0.5)

points <- list()
points[[1]] <- c(pi, 1, 1)

plot_means_hsv(fit1, fit2=fit2, points=points, lines=lines, hsv=TRUE)

# visualize the hue distribution underlying a fit
plot_distributions_hsv(fit1)

# visualize hue distributions underlying two fits
plot_distributions_hsv(fit1, fit2=fit2)

# visualize hue distributions of two fits, add annotation points and lines,
# hsv parameter determines whether annotations are defined in hsv or rgb
lines <- list()
lines[[1]] <- c(2*pi, 1, 1)
lines[[2]] <- c(pi/2, 0.5, 0.5)

points <- list()
points[[1]] <- c(pi, 1, 1)

plot_distributions_hsv(fit1, fit2=fit2, points=points, lines=lines, hsv=TRUE)
```

`compare_distributions` *compare_distributions*

Description

`compare_distributions` draws samples from distribution of the first fit and compares them against samples drawn from the distribution of the second fit, or against samples from multiple fits.

Usage

```
compare_distributions(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?compare_distributions_ttest or ?compare_distributions_linear.

---

compare_distributions,color_class-method
compare_distributions

Description

compare_distributions draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group or against a color defined with rgb or hsv components. You can also provide the rope parameter or execute the comparison only through chosen color components (r, g, b, h, s, v).

Usage

```
## S4 method for signature 'color_class'
compare_distributions(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, rope - region of practical equivalence, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

compare_distributions,linear_class-method
compare_distributions

Description

compare_distributions draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group.

Usage

```
## S4 method for signature 'linear_class'
compare_distributions(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object, rope_intercept and rope_slope - regions of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

compare_distributions,reaction_time_class-method
compare_distributions

Description

compare_distributions draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group or from samples drawn from distributions of multiple groups.

Usage

```
## S4 method for signature 'reaction_time_class'
compare_distributions(object, ...)
```

Arguments

object	reaction_time_class object.
...	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects, rope - region of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

---

compare_distributions,success_rate_class-method
compare_distributions

Description

compare_distributions draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group or from samples drawn from distributions of multiple groups.

Usage

```
## S4 method for signature 'success_rate_class'
compare_distributions(object, ...)
```

Arguments

object	success_rate_class object.
...	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects, rope - region of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

compare_distributions,ttest_class-method *compare_distributions*

Description

compare_distributions draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group, against samples drawn from distributions of multiple groups, against a mean value or against samples from a normal distribution with a defined mean value and variance.

Usage

```
## S4 method for signature 'ttest_class'
compare_distributions(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, fits - a list of ttest_class objects, mu - mean value, sigma - standard deviation, rope - region of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

compare_means *compare_means*

Description

compare_means prints difference between means of two or multiple fits.

Usage

```
compare_means(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?compare_means_ttest or ?compare_means_linear.

---

compare_means,color_class-method
compare_means

Description

compare_means prints difference in colors between two fits or a fit and a color.

Usage

```
## S4 method for signature 'color_class'
compare_means(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, rope - region of practical equivalence, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

compare_means,linear_class-method
compare_means

Description

compare_means prints difference in intercept and slope between two groups.

Usage

```
## S4 method for signature 'linear_class'
compare_means(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object, rope_intercept and rope_slope - regions of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

compare_means,reaction_time_class-method
compare_means

Description

compare_means prints difference in reaction times between two groups or multiple groups.

Usage

```
## S4 method for signature 'reaction_time_class'
compare_means(object, ...)
```

Arguments

object	reaction_time_class object.
...	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects, rope - region of practical equivalence, par - specific parameter of comparison (mu or lambda).

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

---

compare_means,success_rate_class-method
compare_means

Description

compare_means prints difference in success rate between two groups or multiple groups.

Usage

```
## S4 method for signature 'success_rate_class'
compare_means(object, ...)
```

Arguments

object	success_rate_class object.
...	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects, rope - region of practical equivalence.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

compare_means,ttest_class-method *compare_means*

Description

compare_means prints difference/equality of the first group against the second group, against multiple groups, against a mean value or against a normal distribution with a defined mean value and variance.

Usage

```
## S4 method for signature 'ttest_class'
compare_means(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, mu - mean value, sigma - standard deviation, fits - a list of ttest_class objects, rope - region of practical equivalence, par - execute comparison through the sigma or nu parameter.

Value

Comparison results or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

get_parameters *get_parameters*

Description

get_parameters returns a dataframe with values of fitted parameters.

Usage

```
get_parameters(object)
```

Arguments

object	S4 class object from bayes4psy library.

---

get_parameters,color_class-method
get_parameters

Description

get_parameters returns a dataframe with values of fitted parameters.

Usage

```
## S4 method for signature 'color_class'
get_parameters(object)
```

Arguments

object	color_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

---

get_parameters,linear_class-method
get_parameters

Description

get_parameters returns a dataframe with values of fitted parameters.

Usage

```
## S4 method for signature 'linear_class'
get_parameters(object)
```

Arguments

object	linear_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

---

get_parameters,reaction_time_class-method
get_parameters

Description

`get_parameters` returns a dataframe with values of fitted parameters.

Usage

```
## S4 method for signature 'reaction_time_class'
get_parameters(object)
```

Arguments

`object`	reaction_time_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

get_parameters,success_rate_class-method *get_parameters*

Description

get_parameters returns a dataframe with values of fitted parameters.

Usage

```
## S4 method for signature 'success_rate_class'
get_parameters(object)
```

Arguments

object	success_rate_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

get_parameters,ttest_class-method *get_parameters*

Description

get_parameters returns a dataframe with values of fitted parameters.

Usage

```
## S4 method for signature 'ttest_class'
get_parameters(object)
```

Arguments

object	ttest_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

---

get_prior_id *get_prior_id*

Description

`get_prior_id` returns an integer id of prior's family (1 = uniform, 2 = normal, 3 = gamma, 4 = beta).

Usage

```
get_prior_id(object)

## S4 method for signature 'b_prior'
get_prior_id(object)
```

Arguments

`object`	b_prior object.

---

get_subject_parameters *get_subject_parameters*

Description

`get_subject_parameters` returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

Usage

```
get_subject_parameters(object)
```

Arguments

`object`	S4 class object from bayes4psy library.

get_subject_parameters,linear_class-method
get_subject_parameters

Description

get_subject_parameters returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

Usage

```
## S4 method for signature 'linear_class'
get_subject_parameters(object)
```

Arguments

object	linear_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

get_subject_parameters,reaction_time_class-method
get_subject_parameters

Description

get_subject_parameters returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

Usage

```
## S4 method for signature 'reaction_time_class'
get_subject_parameters(object)
```

Arguments

object	reaction_time_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

---

get_subject_parameters,success_rate_class-method
get_subject_parameters

Description

`get_subject_parameters` returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

Usage

```
## S4 method for signature 'success_rate_class'
get_subject_parameters(object)
```

Arguments

`object`	success_rate_class object.

Value

A data frame with parameter values.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

`linear_class-class` *linear_class*

Description

An S4 class for storing results of normal linear model.

Functions

summary(‘linear_class‘): prints a summary of the fit.

print(‘linear_class‘): prints a more detailed summary of the fit

show(‘linear_class‘): prints a more detailed summary of the fit.

plot(‘linear_class‘): plots fitted model against the data. Use this function to explore the quality of your fit. Fit will be plotted on the subject level.

plot(‘linear_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the subjects level (subjects=FALSE).

plot_fit(‘linear_class‘): plots fitted model against the data. Use this function to explore the quality of your fit. Fit will be plotted on the subject level.

plot_fit(‘linear_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the subjects level (subjects=FALSE).

plot_trace(‘linear_class‘): traceplot for main fitted model parameters.

get_parameters(‘linear_class‘): returns a dataframe with values of fitted parameters.

get_subject_parameters(‘linear_class‘): returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

compare_means(‘linear_class‘, fit2=‘linear_class‘): prints difference in slope and intercept between two groups. You can also provide the rope parameter.

plot_means_difference(‘linear_class‘, fit2=‘linear_class‘): a visualization of the difference between two groups. You can plot only slope or intercept by using the par parameter. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_means(‘linear_class‘): plots density of means. You can plot only slope or intercept by using the par parameter.

plot_means(‘linear_class‘, fit2=‘linear_class‘): plots density for the first and the second group means. You can plot only slope or intercept by using the par parameter.

compare_distributions(‘linear_class‘, fit2=‘linear_class‘): draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group.

plot_distributions(‘linear_class‘): a visualization of the fitted distribution.

plot_distributions(‘linear_class‘, fit2=‘linear_class‘): a visualization of two fitted distribution.

plot_distributions_difference(‘linear_class‘, fit2=‘linear_class‘): a visualization of the difference between the distribution of the first group and the second group. You can plot only slope or intercept by using the par parameter. You can also provide the rope and bins (number of bins in the histogram) parameters.

Slots

`extract` Extract from Stan fit.

`fit` Stan fit.

`data` Raw data for the tested group.

Examples

```
# priors
mu_prior <- b_prior(family="normal", pars=c(0, 100))
sigma_prior <- b_prior(family="uniform", pars=c(0, 500))

# attach priors to relevant parameters
priors <- list(c("mu_a", mu_prior),
               c("sigma_a", sigma_prior),
               c("mu_b", mu_prior),
               c("sigma_b", sigma_prior),
               c("mu_s", sigma_prior),
               c("sigma_s", sigma_prior))


# generate data and fit
x <- vector()
y <- vector()
s <- vector()
for (i in 1:5) {
  x <- c(x, rep(1:10, 2))
  y <- c(y, rnorm(20, mean=1:10, sd=2))
  s <- c(s, rep(i, 20))
}

fit1 <- b_linear(x=x, y=y, s=s, priors=priors, chains=1)

fit2 <- b_linear(x=x, y=-2*y, s=s, priors=priors, chains=1)

# a short summary of fitted parameters
summary(fit1)

# a more detailed summary of fitted parameters
print(fit1)
show(fit1)

# plot the fitted distribution against the data
plot(fit1)
plot_fit(fit1)

# plot the fitted distribution against the data,
# plot on the top (group) level
plot(fit1, subjects=FALSE)
plot_fit(fit1, subjects=FALSE)

# traceplot of the fitted parameters
```

```
plot_trace(fit1)
# extract parameter values from the fit
parameters <- get_parameters(fit1)

# extract parameter values on the bottom (subject) level from the fit
subject_parameters <- get_subject_parameters(fit1)

# compare means between two fits
compare_means(fit1, fit2=fit2)

# compare means between two fits, use a rope interval for intercept and slope
compare_means(fit1, fit2=fit2, rope_intercept=0.5, rope_slope=0.2)

# visualize difference in means between two fits
plot_means_difference(fit1, fit2=fit2)

# visualize difference in means between two fits,
# use a rope interval for intercept and slope,
# set the number of bins in the histogram
plot_means_difference(fit1, fit2=fit2, rope_intercept=0.5, rope_slope=0.2, bins=20)

# visualize difference in means between two fits, compare only slope
plot_means_difference(fit1, fit2=fit2, par="slope")

# visualize means of a single fit
plot_means(fit1)

# visualize means of two fits
plot_means(fit1, fit2=fit2)

# visualize means of two fits, plot slope only
plot_means(fit1, fit2=fit2, par="slope")

# draw samples from distributions underlying two fits and compare them,
# use a rope interval for intercept and slope
compare_distributions(fit1, fit2=fit2, rope_intercept=0.5, rope_slope=0.2)

# visualize the distribution underlying a fit
plot_distributions(fit1)

# visualize distributions underlying two fits
plot_distributions(fit1, fit2=fit2)

# visualize distributions underlying two fits, plot slope only
plot_distributions(fit1, fit2=fit2, par="slope")

# visualize difference between distributions underlying two fits
plot_distributions_difference(fit1, fit2=fit2)

# visualize difference between distributions underlying two fits,
# use a rope interval for intercept and slope,
# set the number of bins in the histogram
plot_distributions_difference(fit1, fit2=fit2, rope_intercept=0.5, rope_slope=0.2, bins=20)
```

```
# visualize difference between distributions underlying two fits, plot slope only
plot_distributions_difference(fit1, fit2=fit2, par="slope")
```

`mcmc_hdi` *mcmc_hdi*

Description

A function for calculating the HDI (highest density interval) of a vector of values.

Usage

```
mcmc_hdi(samples, cred_mass = 0.95)
```

Arguments

`samples`	vector of values.
`cred_mass`	credibility mass that the interval should include (default = 0.95).

Value

Boundaries of the HDI.

Author(s)

John Kruschke

`plot,color_class,missing-method` *plot*

Description

`plot` plots fitted model against the data. Use this function to explore the quality of your fit. You can compare fit with underlying data only through chosen color components (r, g, b, h, s, v).

Usage

```
## S4 method for signature 'color_class,missing'
plot(x, y, ...)
```

Arguments

x	color_class object.
y	empty dummy variable, ignore this.
...	pars - components of comparison, a subset of (r, g, b, h, s, v).

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

---

plot,linear_class,missing-method
plot

Description

plot plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'linear_class,missing'
plot(x, y, ...)
```

Arguments

x	linear_class object.
y	empty dummy variable, ignore this.
...	subjects - plot fits on a subject level (default = TRUE).

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

plot,reaction_time_class,missing-method
plot

Description

plot plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'reaction_time_class,missing'
plot(x, y, ...)
```

Arguments

x	reaction_time_class object.
y	empty dummy variable, ignore this.
...	subjects - plot fits on a subject level (default = TRUE).

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

plot,success_rate_class,missing-method
plot

Description

plot plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'success_rate_class,missing'
plot(x, y, ...)
```

Arguments

x	success_rate_class object.
y	empty dummy variable, ignore this.
...	subjects - plot fits on a subject level (default = TRUE).

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

`plot,ttest_class,missing-method` *plot*

Description

`plot` plots fitted model against the data. Use this function to explore the quality of your fit.

Usage

```
## S4 method for signature 'ttest_class,missing'
plot(x)
```

Arguments

`x`	ttest_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

`plot_distributions` *plot_distributions*

Description

`plot_distributions` visualizes fitted distributions.

Usage

```
plot_distributions(object, ...)
```

Arguments

`object`	S4 class object from bayes4psy library.
`...`	see documentation for specific class for the description of available parameters, e.g. ?plot_distributions_ttest or ?plot_distributions_linear.

plot_distributions,color_class-method *plot_distributions*

Description

plot_distributions a visualization of the fitted distributions or constant colors.

Usage

```
## S4 method for signature 'color_class'
plot_distributions(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_distributions,linear_class-method *plot_distributions*

Description

plot_distributions a visualization of the fitted distribution, for one or two fits.

Usage

```
## S4 method for signature 'linear_class'
plot_distributions(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

---

plot_distributions,reaction_time_class-method
plot_distributions

Description

plot_distributions a visualization of the fitted distribution, for one, two or multiple fits.

Usage

```
## S4 method for signature 'reaction_time_class'
plot_distributions(object, ...)
```

Arguments

object	reaction_time_class object.
...	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

plot_distributions,success_rate_class-method *plot_distributions*

Description

plot_distributions a visualization of the fitted distribution, for one, two or multiple fits.

Usage

```
## S4 method for signature 'success_rate_class'
plot_distributions(object, ...)
```

Arguments

object	success_rate_class object.
...	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_distributions,ttest_class-method *plot_distributions*

Description

plot_distributions visualizes distributions underlying tested groups.

Usage

```
## S4 method for signature 'ttest_class'
plot_distributions(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, fits - a list of ttest_class objects, mu - mean value, sigma - standard deviation.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

---

plot_distributions_difference

plot_distributions_difference

Description

plot_distributions_difference a visualization of the difference between the distributions of two or more fits.

Usage

```
plot_distributions_difference(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?plot_distributions_difference_ttest or ?plot_distributions_difference_linear.

plot_distributions_difference,color_class-method
plot_distributions_difference

Description

plot_distributions_difference a visualization of the difference between the distribution of the first group and the second group.

Usage

```
## S4 method for signature 'color_class'
plot_distributions_difference(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, rope - region of practical equivalence, bins - number of bins in the histogram, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_distributions_difference,linear_class-method
plot_distributions_difference

Description

plot_distributions_difference visualizes the difference between two groups.

Usage

```
## S4 method for signature 'linear_class'
plot_distributions_difference(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object, par - specific parameter of comparison (slope or intercept), rope_intercept and rope_slope - regions of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

---

plot_distributions_difference,reaction_time_class-method
plot_distributions_difference

Description

plot_distributions_difference a visualization of the difference between the distribution of the first group and the second group or between multiple groups.

Usage

```
## S4 method for signature 'reaction_time_class'
plot_distributions_difference(object, ...)
```

Arguments

object	reaction_time_class object.
...	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects, rope - region of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

`plot_distributions_difference,success_rate_class-method`
plot_distributions_difference

Description

`plot_distributions_difference` a visualization of the difference between the distribution of the first group and the second group or between multiple groups.

Usage

```
## S4 method for signature 'success_rate_class'
plot_distributions_difference(object, ...)
```

Arguments

`object`	success_rate_class object.
`...`	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects, rope - region of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_distributions_difference,ttest_class-method
plot_distributions_difference

Description

plot_distributions_difference a visualization of the difference between the distribution of the first group, the distribution or a constant value for the second group or between multiple distributions.

Usage

```
## S4 method for signature 'ttest_class'
plot_distributions_difference(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, fits - a list of ttest_class objects, mu - mean value, sigma - standard deviation, rope - region of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

plot_distributions_hsv
plot_distributions_hsv

Description

plot_distributions_hsv a visualization of distributions of one or two fits thorough a circular visualization of hsv color components. You can also compare fit means with colors defined through rgb or hsv components (as points or as lines on the visualization).

Usage

```
plot_distributions_hsv(object, ...)

## S4 method for signature 'color_class'
plot_distributions_hsv(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, points - points to plot defined through rgb or hsv, lines - lines to plot defined through rgb or hsv, hsv - are points and lines defined in hsv format (default = FALSE).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

---

plot_fit *plot_fit*

Description

`plot_fit` plots fitted model against the data. Use this function to explore the quality of your fit.

Usage

```
plot_fit(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?plot_fit_colors or ?plot_fit_linear.

plot_fit,color_class-method
plot_fit

Description

plot_fit plots fitted model against the data. Use this function to explore the quality of your fit. You can compare fit with underlying data only through chosen color components (r, g, b, h, s, v).

Usage

```
## S4 method for signature 'color_class'
plot_fit(object, ...)
```

Arguments

object	color_class object.
...	pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_fit,linear_class-method
plot_fit

Description

plot_fit plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'linear_class'
plot_fit(object, ...)
```

Arguments

object	linear_class object.
...	subjects - plot fits on a subject level (default = TRUE).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

---

plot_fit,reaction_time_class-method
plot_fit

Description

plot_fit plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'reaction_time_class'
plot_fit(object, ...)
```

Arguments

object	reaction_time_class object.
...	subjects - plot fits on a subject level (default = TRUE).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

plot_fit,success_rate_class-method
plot_fit

Description

plot_fit plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

Usage

```
## S4 method for signature 'success_rate_class'
plot_fit(object, ...)
```

Arguments

object	success_rate_class object.
...	subjects - plot fits on a subject level (default = TRUE).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_fit,ttest_class-method
plot_fit

Description

plot_fit plots fitted model against the data. Use this function to explore the quality of your fit.

Usage

```
## S4 method for signature 'ttest_class'
plot_fit(object)
```

Arguments

object	ttest_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

---

plot_fit_hsv *plot_fit_hsv*

Description

plot_fit_hsv plots fitted model against the data. Use this function to explore the quality of your fit thorough a circular visualization of hsv color components.

Usage

```
plot_fit_hsv(object)

## S4 method for signature 'color_class'
plot_fit_hsv(object)
```

Arguments

object	color_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_hsv *plot_hsv*

Description

plot_hsv plots fitted model against the data. Use this function to explore the quality of your fit thorough a circular visualization of hsv color components.

Usage

```
plot_hsv(object)

## S4 method for signature 'color_class'
plot_hsv(object)
```

Arguments

object	color_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_means *plot_means*

Description

plot_means plots means for one or multiple fits.

Usage

```
plot_means(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?plot_means_ttest or ?plot_means_linear.

plot_means,color_class-method
plot_means

Description

plot_means plots density of means, the first and the second group means or a constant values in case second group is defined as rgb or hsv color.

Usage

```
## S4 method for signature 'color_class'
plot_means(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

plot_means,linear_class-method
plot_means

Description

plot_means plots means or the first and the second group means.

Usage

```
## S4 method for signature 'linear_class'
plot_means(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object, par - plot a specific parameter (slope or intercept).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

---

plot_means,reaction_time_class-method
plot_means

Description

plot_means plots density of means for one, two or multiple groups.

Usage

```
## S4 method for signature 'reaction_time_class'
plot_means(object, ...)
```

Arguments

object	reaction_time_class object.
...	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects, par - plot a specific parameter (mu or lambda).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

plot_means,success_rate_class-method
plot_means

Description

plot_means plots density of means for one, two or multiple groups.

Usage

```
## S4 method for signature 'success_rate_class'
plot_means(object, ...)
```

Arguments

object	success_rate_class object.
...	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_means,ttest_class-method
plot_means

Description

plot_means plots density of means, the first and the second group means, means of multiple groups or a mean value in case second group is defined as a constant.

Usage

```
## S4 method for signature 'ttest_class'
plot_means(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, mu - mean value, fits - a list of ttest_class objects, par - plot the sigma or nu parameter.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

---

plot_means_difference *plot_means_difference*

Description

plot_means_difference plots difference between means of two or multiple fits.

Usage

```
plot_means_difference(object, ...)
```

Arguments

object	S4 class object from bayes4psy library.
...	see documentation for specific class for the description of available parameters, e.g. ?plot_means_difference_ttest or ?plot_means_difference_linear.

---

plot_means_difference,color_class-method *plot_means_difference*

Description

plot_means_difference a visualization of the difference between two fits

Usage

```
## S4 method for signature 'color_class'
plot_means_difference(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, rgb - color defined through rgb, hsv - color defined through rgb, rope - region of practical equivalence, bins - number of bins in the histogram, pars - components of comparison, a subset of (r, g, b, h, s, v).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

---

plot_means_difference,linear_class-method
plot_means_difference

Description

plot_means_difference plots difference between two groups.

Usage

```
## S4 method for signature 'linear_class'
plot_means_difference(object, ...)
```

Arguments

object	linear_class object.
...	fit2 - a second linear_class object, par - specific parameter of comparison (slope or intercept), rope_intercept and rope_slope - regions of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

`plot_means_difference,reaction_time_class-method`
plot_means_difference

Description

`plot_means_difference` a visualization of the difference between two groups or multiple groups.

Usage

```
## S4 method for signature 'reaction_time_class'
plot_means_difference(object, ...)
```

Arguments

`object`	reaction_time_class object.
`...`	fit2 - a second reaction_time_class object, fits - a list of reaction_time_class objects, rope - region of practical equivalence, bins - number of bins in the histogram, par - specific parameter of comparison (mu or lambda).

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

plot_means_difference,success_rate_class-method
plot_means_difference

Description

plot_means_difference a visualization of the difference between two groups or multiple groups.

Usage

```
## S4 method for signature 'success_rate_class'
plot_means_difference(object, ...)
```

Arguments

object	success_rate_class object.
...	fit2 - a second success_rate_class object, fits - a list of success_rate_class objects, rope - region of practical equivalence, bins - number of bins in the histogram.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_means_difference,ttest_class-method
plot_means_difference

Description

plot_means_difference a visualization of the difference of the first group against the second group, against multiple groups, against a mean value or against a normal distribution with a defined mean value and variance.

Usage

```
## S4 method for signature 'ttest_class'
plot_means_difference(object, ...)
```

Arguments

object	ttest_class object.
...	fit2 - a second ttest_class object, fits - a list of ttest_class objects, mu - mean value, rope - region of practical equivalence, bins - number of bins in the histogram, par - compare through the sigma or nu parameter.

Value

A ggplot visualization or an error if something went wrong.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

---

plot_means_hsv *plot_means_hsv*

Description

plot_means_hsv a visualization of the difference between means of two fits through a circular visualization of hsv color components. You can also compare fit means with colors defined through rgb or hsv components (as points or as lines on the visualization).

Usage

```
plot_means_hsv(object, ...)

## S4 method for signature 'color_class'
plot_means_hsv(object, ...)
```

Arguments

object	color_class object.
...	fit2 - a second color_class object, points - points to plot defined through rgb or hsv, lines - lines to plot defined through rgb or hsv, hsv - are points and lines defined in hsv format (default = FALSE).

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

`plot_trace` *plot_trace*

Description

`plot_trace` traceplot for main fitted model parameters.

Usage

```
plot_trace(object)
```

Arguments

`object`	S4 class object from bayes4psy library.

`plot_trace,color_class-method` *plot_trace*

Description

`plot_trace` traceplot for main fitted model parameters.

Usage

```
## S4 method for signature 'color_class'
plot_trace(object)
```

Arguments

`object`	color_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

`plot_trace,linear_class-method`
plot_trace

Description

`plot_trace` traceplot for main fitted model parameters.

Usage

```
## S4 method for signature 'linear_class'
plot_trace(object)
```

Arguments

`object`	linear_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

`plot_trace,reaction_time_class-method`
plot_trace

Description

`plot_trace` traceplot for main fitted model parameters.

Usage

```
## S4 method for signature 'reaction_time_class'
plot_trace(object)
```

Arguments

object	reaction_time_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

---

plot_trace,success_rate_class-method
plot_trace

Description

plot_trace traceplot for main fitted model parameters.

Usage

```
## S4 method for signature 'success_rate_class'
plot_trace(object)
```

Arguments

object	success_rate_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

plot_trace,ttest_class-method *plot_trace*

Description

plot_trace traceplot for main fitted model parameters.

Usage

```
## S4 method for signature 'ttest_class'
plot_trace(object)
```

Arguments

object	ttest_class object.

Value

A ggplot visualization.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

reaction_time_class-class *reaction_time_class*

Description

An S4 class for storing results of reaction time Bayesian model.

Functions

summary(‘reaction_time_class‘): prints a summary of the fit.

print(‘reaction_time_class‘): prints a more detailed summary of the fit

show(‘reaction_time_class‘): prints a more detailed summary of the fit.

plot(‘reaction_time_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot(‘reaction_time_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the group level (subjects=FALSE).

plot_fit(‘reaction_time_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot_fit(‘reaction_time_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subject level (subjects=TRUE) or on the group level (subjects=FALSE).

plot_trace(‘reaction_time_class‘): traceplot for main fitted model parameters.

get_parameters(‘reaction_time_class‘): returns a dataframe with values of fitted parameters.

get_subject_parameters(‘reaction_time_class‘): returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

compare_means(‘reaction_time_class‘, fit2=‘reaction_time_class‘): returns difference in reaction times between two groups. You can also provide the rope parameter or execute the comparison only through a chosen parameter - mu or lambda.

compare_means(‘reaction_time_class‘, fits=‘list‘): returns difference in reaction times between multiple groups. You can also provide the rope parameter. You can also provide the rope parameter or execute the comparison only through a chosen parameter - mu or lambda.

plot_means_difference(‘reaction_time_class‘, fit2=‘reaction_time_class‘): a visualization of the difference between two groups. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through a chosen parameter - mu or lambda.

plot_means_difference(‘reaction_time_class‘, fits=‘list‘): a visualization of the difference between multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison only through a chosen parameter - mu or lambda.

plot_means(‘reaction_time_class‘): plots density of the means. You can also visualize the density only for a chosen parameter - mu or lambda.

plot_means(‘reaction_time_class‘, fit2=‘reaction_time_class‘): plots density for the first and the second group means. You can also visualize the density only for a chosen parameter - mu or lambda.

plot_means(‘reaction_time_class‘, fits=‘list‘): plots density for means of multiple groups. You can also visualize the density only for a chosen parameter - mu or lambda.

compare_distributions(‘reaction_time_class‘, fit2=‘reaction_time_class‘): draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group. You can also provide the rope parameter.

compare_distributions(‘reaction_time_class‘, fits=‘lists‘): draws and compares samples from distributions of multiple groups. You can also provide the rope parameter.

plot_distributions(‘reaction_time_class‘): a visualization of the fitted distribution.

plot_distributions(‘reaction_time_class‘, fit2=‘reaction_time_class‘): a visualization of the distribution for two fits.

plot_distributions(‘reaction_time_class‘, fits=‘list‘): a visualization of the distribution for multiple fits.

plot_distributions_difference(‘reaction_time_class‘, fit2=‘reaction_time_class‘): a visualization of the difference between the distribution of the first group and the second group. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_distributions_difference(‘reaction_time_class‘, fits=‘list‘): a visualization of the difference between the distributions of multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters.

Slots

`extract` Extract from Stan fit.

`fit` Stan fit.

`data` Data on which the fit is based.

Examples

```
# priors
mu_prior <- b_prior(family="normal", pars=c(0, 100))
sigma_prior <- b_prior(family="uniform", pars=c(0, 500))
lambda_prior <- b_prior(family="uniform", pars=c(0.05, 5))

# attach priors to relevant parameters
priors <- list(c("mu_m", mu_prior),
               c("sigma_m", sigma_prior),
               c("mu_s", sigma_prior),
               c("sigma_s", sigma_prior),
               c("mu_l", lambda_prior),
               c("sigma_l", sigma_prior))


# subjects
s <- rep(1:5, 20)

# generate data and fit
rt1 <- emg::remg(100, mu=10, sigma=1, lambda=0.4)
fit1 <- b_reaction_time(t=rt1, s=s, priors=priors, chains=1)

rt2 <- emg::remg(100, mu=10, sigma=2, lambda=0.1)
fit2 <- b_reaction_time(t=rt2, s=s, priors=priors, chains=1)

rt3 <- emg::remg(100, mu=20, sigma=2, lambda=1)
fit3 <- b_reaction_time(t=rt3, s=s, priors=priors, chains=1)

rt4 <- emg::remg(100, mu=15, sigma=2, lambda=0.5)
fit4 <- b_reaction_time(t=rt4, s=s, priors=priors, chains=1)

# fit list
fit_list <- list(fit2, fit3, fit4)

# a short summary of fitted parameters
summary(fit1)

# a more detailed summary of fitted parameters
print(fit1)
show(fit1)

# plot the fitted distribution against the data
plot(fit1)
plot_fit(fit1)
```

```
# plot the fitted distribution against the data,
# plot on the top (group) level
plot(fit1, subjects=FALSE)
plot_fit(fit1, subjects=FALSE)

# traceplot of the fitted parameters
plot_trace(fit1)

# extract parameter values from the fit
parameters <- get_parameters(fit1)

# extract parameter values on the bottom (subject) level from the fit
subject_parameters <- get_subject_parameters(fit1)

# compare means between two fits, use a rope interval
compare_means(fit1, fit2=fit2, rope=0.5)

# compare means between two fits,
# use only the mu parameter of the exponentially modified gaussian distribution
compare_means(fit1, fit2=fit2, par="mu")

# compare means between multiple fits
compare_means(fit1, fits=fit_list)

# visualize difference in means between two fits,
# specify number of histogram bins and rope interval
plot_means_difference(fit1, fit2=fit2, bins=20, rope=0.5)

# visualize difference in means between two fits,
# use only the mu parameter of the exponentially modified gaussian distribution
plot_means_difference(fit1, fit2=fit2, par="mu")

# visualize difference in means between multiple fits
plot_means_difference(fit1, fits=fit_list)

# visualize means of a single fit
plot_means(fit1)

# visualize means of two fits
plot_means(fit1, fit2=fit1)

# visualize means of two fits,
# use only the mu parameter of the exponentially modified gaussian distribution
plot_means(fit1, fit2=fit2, par="mu")

# visualize means of multiple fits
plot_means(fit1, fits=fit_list)

# draw samples from distributions underlying two fits and compare them,
# use a rope interval
compare_distributions(fit1, fit2=fit2, rope=0.5)

# draw samples from distributions underlying multiple fits and compare them
```

```
compare_distributions(fit1, fits=fit_list)

# visualize the distribution underlying a fit
plot_distributions(fit1)

# visualize distributions underlying two fits
plot_distributions(fit1, fit2=fit2)

# visualize distributions underlying multiple fits
plot_distributions(fit1, fits=fit_list)

# visualize difference between distributions underlying two fits,
# use a rope interval
plot_distributions_difference(fit1, fit2=fit2, rope=0.05)

# visualize difference between distributions underlying multiple fits
plot_distributions_difference(fit1, fits=fit_list)
```

`show,color_class-method` *show*

Description

show prints a more detailed summary of the Bayesian color fit.

Usage

```
## S4 method for signature 'color_class'
show(object)
```

Arguments

object	color_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

show,linear_class-method
show

Description

show prints a more detailed summary of the Bayesian linear model fit.

Usage

```
## S4 method for signature 'linear_class'
show(object)
```

Arguments

object	linear_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

show,reaction_time_class-method
show

Description

show prints a more detailed summary of the Bayesian reaction time fit.

Usage

```
## S4 method for signature 'reaction_time_class'
show(object)
```

Arguments

object	reaction_time_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

show,success_rate_class-method *show*

Description

show prints a more detailed summary of the Bayesian success rate fit.

Usage

```
## S4 method for signature 'success_rate_class'
show(object)
```

Arguments

object	success_rate_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

show,ttest_class-method *show*

Description

show prints a more detailed summary of the Bayesian ttest fit.

Usage

```
## S4 method for signature 'ttest_class'
show(object)
```

Arguments

object	ttest_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

`success_rate_class-class` *success_rate_class*

Description

An S4 class for storing results of successes (true/false) Bayesian model.

Functions

summary(‘success_rate_class‘): prints a summary of the fit.

print(‘success_rate_class‘): prints a more detailed summary of the fit

show(‘success_rate_class‘): prints a more detailed summary of the fit.

plot(‘success_rate_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot(‘success_rate_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

plot_fit(‘success_rate_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot_fit(‘success_rate_class‘, subjects=’boolean’): plots fitted model against the data. Use this function to explore the quality of your fit. You can plot on the subjects level (subjects=TRUE) or on the group level (subjects=FALSE).

plot_trace(‘success_rate_class‘): traceplot for main fitted model parameters.

get_parameters(‘success_rate_class‘): returns a dataframe with values of fitted parameters.

get_subject_parameters(‘success_rate_class‘): returns a dataframe with values of fitted parameters for each subject in the hierarchical model.

compare_means(‘success_rate_class‘, fit2=‘success_rate_class‘): returns difference in success rate between two groups. You can also provide the rope parameter.

compare_means(‘success_rate_class‘, fits=‘list‘): returns difference in success rate between multiple groups. You can also provide the rope parameter.

plot_means_difference(‘success_rate_class‘, fit2=‘success_rate_class‘): a visualization of the difference between two groups. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_means_difference(‘success_rate_class‘, fits=‘list‘): a visualization of the difference between multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_means(‘success_rate_class‘): plots density for the first group means.

plot_means(‘success_rate_class‘, fit2=‘success_rate_class‘): plots density for the first and the second group means.

plot_means(‘success_rate_class‘, fits=‘list‘): plots density for multiple

compare_distributions(‘success_rate_class‘, fit2=‘success_rate_class‘): draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group. You can also provide the rope parameter.

compare_distributions(‘success_rate_class‘, fits=‘list‘): draws and compares samples from distributions of multiple groups. You can also provide the rope parameter.

plot_distributions(‘success_rate_class‘): a visualization of the fitted distribution.

plot_distributions(‘success_rate_class‘, fit2=‘success_rate_class‘): a visualization of the distribution for two fits.

plot_distributions(‘success_rate_class‘, fits=‘list‘): a visualization of the distribution for multiple fits.

plot_distributions_difference(‘success_rate_class‘, fit2=‘success_rate_class‘): a visualization of the difference between the distribution of the first group and the second group. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_distributions_difference(‘success_rate_class‘, fits=‘list‘): a visualization of the difference between the distributions of multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_fit(‘success_rate_class‘): plots fitted model against the data. Use this function to explore the quality of your fit. Fit will be plotted on the group level.

Slots

`extract` Extract from Stan fit.

`fit` Stan fit.

`data` Data on which the fit is based.

Examples

```
# priors
p_prior <- b_prior(family = "beta", pars = c(1, 1))
tau_prior <- b_prior(family = "uniform", pars = c(0, 500))

# attach priors to relevant parameters
priors <- list(
  c("p", p_prior),
  c("tau", tau_prior)
)

# subjects
s <- rep(1:5, 20)

# generate data and fit
data1 <- rbinom(100, size = 1, prob = 0.6)
fit1 <- b_success_rate(r = data1, s = s, priors = priors, chains = 1)

data2 <- rbinom(100, size = 1, prob = 0.1)
fit2 <- b_success_rate(r = data2, s = s, priors = priors, chains = 1)

data3 <- rbinom(100, size = 1, prob = 0.5)
```

```
fit3 <- b_success_rate(r = data3, s = s, priors = priors, chains = 1)

data4 <- rbinom(100, size = 1, prob = 0.9)
fit4 <- b_success_rate(r = data4, s = s, priors = priors, chains = 1)

# fit list
fit_list <- list(fit2, fit3, fit4)

# a short summary of fitted parameters
summary(fit1)

# a more detailed summary of fitted parameters
print(fit1)
show(fit1)

# plot the fitted distribution against the data
plot(fit1)
plot_fit(fit1)

# plot the fitted distribution against the data,
# plot on the top (group) level
plot(fit1, subjects = FALSE)
plot_fit(fit1, subjects = FALSE)

# traceplot of the fitted parameters
plot_trace(fit1)

# extract parameter values from the fit
parameters <- get_parameters(fit1)

# extract parameter values on the bottom (subject) level from the fit
subject_parameters <- get_subject_parameters(fit1)

# compare means between two fits, use a rope interval
compare_means(fit1, fit2 = fit2, rope = 0.05)

# compare means between multiple fits
compare_means(fit1, fits = fit_list)

# visualize difference in means between two fits,
# specify number of histogram bins and rope interval
plot_means_difference(fit1, fit2 = fit2, bins = 40, rope = 0.05)

# visualize difference in means between multiple fits
plot_means_difference(fit1, fits = fit_list)

# visualize means of a single fit
plot_means(fit1)

# visualize means of two fits
plot_means(fit1, fit2 = fit2)

# visualize means of multiple fits
```

```
plot_means(fit1, fits = fit_list)

# draw samples from distributions underlying two fits and compare them,
# use a rope interval
compare_distributions(fit1, fit2 = fit2, rope = 0.05)

# draw samples from distributions underlying multiple fits and compare them
compare_distributions(fit1, fits = fit_list)

# visualize the distribution underlying a fit
plot_distributions(fit1)

# visualize distributions underlying two fits
plot_distributions(fit1, fit2 = fit2)

# visualize distributions underlying multiple fits
plot_distributions(fit1, fits = fit_list)

# visualize difference between distributions underlying two fits,
# use a rope interval
plot_distributions_difference(fit1, fit2 = fit2, rope = 0.05)

# visualize difference between distributions underlying multiple fits
plot_distributions_difference(fit1, fits = fit_list)
```

---

summary,color_class-method
summary

Description

summary prints summary of the Bayesian color fit.

Usage

```
## S4 method for signature 'color_class'
summary(object)
```

Arguments

object	color_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?color_class
```

summary,linear_class-method
summary

Description

summary prints a summary of the Bayesian linear model fit.

Usage

```
## S4 method for signature 'linear_class'
summary(object)
```

Arguments

object	linear_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?linear_class
```

summary,reaction_time_class-method
summary

Description

summary prints a summary of the Bayesian reaction time fit.

Usage

```
## S4 method for signature 'reaction_time_class'
summary(object)
```

Arguments

object	reaction_time_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?reaction_time_class
```

summary,success_rate_class-method
summary

Description

summary prints a summary of the Bayesian success rate fit.

Usage

```
## S4 method for signature 'success_rate_class'
summary(object)
```

Arguments

object	success_rate_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?success_rate_class
```

summary,ttest_class-method
summary

Description

summary prints a summary of the Bayesian ttest fit.

Usage

```
## S4 method for signature 'ttest_class'
summary(object)
```

Arguments

object	ttest_class object.

Examples

```
# to use the function you first have to prepare the data and fit the model
# see class documentation for an example of the whole process
# along with an example of how to use this function
?ttest_class
```

ttest_class-class *ttest_class*

Description

An S4 class for storing results of Bayesian t-test results.

Functions

summary(‘ttest_class‘): prints a summary of the fit.

print(‘ttest_class‘): prints a more detailed summary of the fit

show(‘ttest_class‘): prints a more detailed summary of the fit.

plot(‘ttest_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot_fit(‘ttest_class‘): plots fitted model against the data. Use this function to explore the quality of your fit.

plot_trace(‘ttest_class‘): traceplot for main fitted model parameters.

get_parameters(‘ttest_class‘): returns a dataframe with values of fitted parameters.

compare_means(‘ttest_class‘, fit2=‘ttest_class‘): prints difference/equality of the first group against the second group. You can also provide the rope parameter or execute the comparison through the sigma or nu parameter.

compare_means(‘ttest_class‘, mu=‘numeric‘): prints difference/equality of the first group against a mean value. You can also provide the rope parameter or execute the comparison through the sigma parameter.

compare_means(‘ttest_class‘, mu=‘numeric‘, sigma=‘numeric‘): prints difference/equality of the first group against a normal distribution provided with mean value and standard deviation. Note here that sigma is used only in the Cohen’s d calculation. You can also provide the rope parameter or execute the comparison through the sigma or nu parameter.

compare_means(‘ttest_class‘, fits=‘list‘): prints difference/equality of the first group and multiple other groups. You can also provide the rope parameter or execute the comparison through the sigma or nu parameter.

plot_means_difference(‘ttest_class‘, fit2=‘ttest_class‘): a visualization of the difference between the first group and the second group. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison through the sigma or nu parameter.

plot_means_difference(‘ttest_class‘, mu=‘numeric‘): a visualization of the difference between the first group and a constant value or a normal distribution with mean value mu. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison through the sigma or nu parameter.

plot_means_difference(‘ttest_class‘, fits=‘list‘): a visualization of the difference between multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters or visualize the comparison through the sigma or nu parameter.

plot_means(‘ttest_class‘): plots density of means. You can also visualize the density for the sigma or nu parameter.

plot_means(‘ttest_class‘, fit2=‘ttest_class‘): plots density for the first and the second group means. You can also visualize the density for the sigma or nu parameter.

plot_means(‘ttest_class‘, mu=‘numeric‘): plots density for the first group means and a mean value in case second group is defined as a normal distribution or as a constant. You can also visualize the density for the sigma or nu parameter.

plot_means(‘ttest_class‘, fits=‘list‘): plots density for the first group means and means for multiple other groups. You can also visualize the density for the sigma or nu parameter.

compare_distributions(‘ttest_class‘, fit2=‘ttest_class‘): draws samples from distribution of the first group and compares them against samples drawn from the distribution of the second group. You can also provide the rope parameter.

compare_distributions(‘ttest_class‘, mu=‘numeric‘): draws samples from distribution of the first group and compares them against a mean value. You can also provide the rope parameter.

compare_distributions(‘ttest_class‘, mu=‘numeric‘, sigma=‘numeric‘): draws samples from distribution of the first group and compares them against samples from a normal distribution with a defined mean value and variance. You can also provide the rope parameter.

compare_distributions(‘ttest_class‘, fits=‘list‘): draws samples from distribution of the first group and compares them against samples drawn from multiple other groups. You can also provide the rope parameter.

plot_distributions(‘ttest_class‘): a visualization of the fitted distribution.

plot_distributions(‘ttest_class‘, fit2=‘ttest_class‘): a visualization of two fitted distributions.

plot_distributions(‘ttest_class‘, mu=‘numeric‘): a visualization of the fitted distribution and a constant value.

plot_distributions(‘ttest_class‘, mu=‘numeric‘, sigma=‘numeric‘): a visualization of the fitted distribution and the normal distribution defined with a mean value and a standard deviation.

plot_distributions(‘ttest_class‘, fits=‘list‘): a visualization of multiple fitted distributions.

plot_distributions_difference(‘ttest_class‘, fit2=‘ttest_class‘): a visualization of the difference between the distribution of the first group and the distribution of the second group. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_distributions_difference(‘ttest_class‘, mu=‘numeric‘): a visualization of the difference between the distribution of the first group and a constant value. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_distributions_difference(‘ttest_class‘, mu=‘numeric‘, sigma=‘numeric‘): a visualization of the difference between the distribution of the first group and the normal distribution defined with a mean value and standard deviation. You can also provide the rope and bins (number of bins in the histogram) parameters.

plot_distributions_difference(‘ttest_class‘, fits=‘list‘): a visualization of the difference between multiple groups. You can also provide the rope and bins (number of bins in the histogram) parameters.

Slots

`extract` Extract from Stan fit.

`fit` Stan fit.

`data` Raw data for the tested group.

Examples

```
# priors
mu_prior <- b_prior(family = "normal", pars = c(0, 1000))
sigma_prior <- b_prior(family = "uniform", pars = c(0, 500))
nu_prior <- b_prior(family = "normal", pars = c(2000, 1000))

# attach priors to relevant parameters
priors <- list(
  c("mu", mu_prior),
  c("sigma", sigma_prior),
  c("nu", nu_prior)
)

# generate data and fit
data1 <- rnorm(20, mean = 150, sd = 20)
fit1 <- b_ttest(data = data1, priors = priors, chains = 1)

data2 <- rnorm(20, mean = 200, sd = 20)
fit2 <- b_ttest(data = data2, priors = priors, chains = 1)

data3 <- rnorm(20, mean = 150, sd = 40)
fit3 <- b_ttest(data = data3, priors = priors, chains = 1)

data4 <- rnorm(20, mean = 50, sd = 10)
fit4 <- b_ttest(data = data4, priors = priors, chains = 1)

# fit list
fit_list <- list(fit2, fit3, fit4)

# a short summary of fitted parameters
summary(fit1)

# a more detailed summary of fitted parameters
print(fit1)
show(fit1)

# plot the fitted distribution against the data
plot(fit1)
plot_fit(fit1)

# traceplot of the fitted parameters
plot_trace(fit1)

# extract parameter values from the fit
parameters <- get_parameters(fit1)

# compare means between two fits
compare_means(fit1, fit2 = fit2)

# compare means between two fits, use a rope interval
compare_means(fit1, fit2 = fit2, rope = 2)
```

```
# compare means between a fit and a constant value
compare_means(fit1, mu = 150)

# compare means between a fit and a distribution,
# sigma is used for calculating Cohen's d
compare_means(fit1, mu = 150, sigma = 20)

# compare means between multiple fits
compare_means(fit1, fits = fit_list)

# visualize difference in means between two fits,
# specify number of histogram bins
plot_means_difference(fit1, fit2 = fit2, bins = 20)

# visualize difference in means between a fit and a constant value
plot_means_difference(fit1, mu = 150)

# visualize difference in means between multiple fits, use a rope interval
plot_means_difference(fit1, fits = fit_list, rope = 2)

# visualize means of a single fit
plot_means(fit1)

# visualize means of two fits
plot_means(fit1, fit2 = fit2)

# visualize means of a fit and a constant value
plot_means(fit1, mu = 150)

# visualize means of multiple fits
plot_means(fit1, fits = fit_list)

# draw samples from distributions underlying two fits and compare them
compare_distributions(fit1, fit2 = fit2)

# draw samples from a distribution underlying the fit
# and compare them with a constant, use a rope interval
compare_distributions(fit1, mu = 150, rope = 2)

# draw samples from a distribution underlying the fit and
# compare them with a user defined distribution
compare_distributions(fit1, mu = 150, sigma = 20)

# draw samples from distributions underlying multiple fits and compare them
compare_distributions(fit1, fits = fit_list)

# visualize the distribution underlying a fit
plot_distributions(fit1)

# visualize distributions underlying two fits
plot_distributions(fit1, fit2 = fit2)

# visualize the distribution underlying a fit and a constant value
```

```
plot_distributions(fit1, mu = 150)

# visualize the distribution underlying a fit and a user defined distribution
plot_distributions(fit1, mu = 150, sigma = 20)

# visualize distributions underlying multiple fits
plot_distributions(fit1, fits = fit_list)

# visualize difference between distributions underlying two fits,
# use a rope interval
plot_distributions_difference(fit1, fit2 = fit2, rope = 2)

# visualize difference between a distribution underlying the fit
# and a constant value
plot_distributions_difference(fit1, mu = 150)

# visualize difference between a distribution underlying the fits
# and a user defined distribution
plot_distributions_difference(fit1, mu = 150, sigma = 20)

# visualize difference between distributions underlying multiple fits
plot_distributions_difference(fit1, fits = fit_list)
```

Index