Package 'RoBSA'

November 8, 2026

Type Package

Title Robust Bayesian Survival Analysis

Version 1.1.0

Maintainer František Bartoš <f.bartos96@gmail.com>

Description A framework for estimating ensembles of parametric survival models with different parametric families. The RoBSA framework uses Bayesian model-averaging to combine the competing parametric survival models into a model ensemble, weights the posterior parameter distributions based on posterior model probabilities and uses Bayes factors to test for the presence or absence of the individual predictors or preference for a parametric family (Bartoš, Aust & Haaf, 2022, <doi:10.1186/s12874-022-01676-9>). The user can define a wide range of informative priors for all parameters of interest. The package provides convenient functions for summary, visualizations, fit diagnostics, and prior distribution calibration.

URL https://fbartos.github.io/RoBSA/

BugReports https://github.com/FBartos/RoBSA/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

SystemRequirements JAGS >= 4.3.1 (https://mcmc-jags.sourceforge.io/)

Depends R (>= 4.0.0)

Imports BayesTools (>= 0.4.0), survival, rjags, runjags, scales, coda, stats, graphics, rlang, Rdpack

Suggests parallel, ggplot2, flexsurv, testthat, vdiffr, knitr, rmarkdown, covr

RdMacros Rdpack

NeedsCompilation yes

Author František Bartoš [aut, cre] (ORCID: <https://orcid.org/0000-0002-0018-5573>), Julia M. Haaf [ths] (ORCID: <https://orcid.org/0000-0001-5122-706X>),

Matthew Denwood [cph] (Original copyright holder of some modified code where indicated.),
Martyn Plummer [cph] (Original copyright holder of some modified code where indicated.)

Repository CRAN

Date/Publication 2026-11-07 23:30:02 UTC

Contents

RoBSA-package . . . 3
calibrate_meta_analytic . . . 3
calibrate_quartiles . . . 4
check_RoBSA . . . 5
check_setup . . . 5
contr.BayesTools . . . 8
default_prior . . . 9
diagnostics . . . 10
exp-aft . . . 11
extract_flexsurv . . . 12
gamma-aft . . . 13
is.RoBSA . . . 14
llogis-aft . . . 14
lnorm-aft . . . 15
plot.RoBSA . . . 16
plot_models . . . 18
plot_prediction . . . 20
predict.RoBSA . . . 22
print.RoBSA . . . 24
print.summary.RoBSA . . . 24
prior . . . 25
prior_factor . . . 26
prior_informed . . . 28
prior_informed_medicine_names . . . 30
prior_none . . . 30
RoBSA . . . 31
RoBSA_control . . . 34
RoBSA_options . . . 35
summary.RoBSA . . . 36
update.RoBSA . . . 38
weibull-aft . . . 41

Index **43**

RoBSA-package *RoBSA: Robust Bayesian survival analysis*

Description

Bayesian model-averaged parametric survival analysis with ability to specify informed prior distributions and draw inference with inclusion Bayes factors. See Bartoš et al. (2022) for more details about the methodology.

User guide

See Bartoš et al. (2022), for details regarding the RoBSA methodology.

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Bartoš F, Aust F, Haaf JM (2022). "Informed Bayesian survival analysis." *BMC Medical Research Methodology*. doi:10.1186/s12874022016769.

See Also

Useful links:

- https://fbartos.github.io/RoBSA/
- Report bugs at https://github.com/FBartos/RoBSA/issues

calibrate_meta_analytic *Create meta-analytic predictive prior distributions*

Description

Calibrates prior distributions for parametric survival analysis based on historical data. Returns a list of prior distribution for the intercepts and auxiliary parameters.

Usage

```
calibrate_meta_analytic(
  datasets,
 distributions = c("exp-aft", "weibull-aft", "lnorm-aft", "llogis-aft", "gamma-aft"),
  prior_mu = prior("cauchy", parameters = list(location = 0, scale = 100)),
 prior_tau = prior("cauchy", parameters = list(location = 0, scale = 10), truncation =
    list(0, Inf)),
  ...
)
```

Arguments

datasets	list of data.frames containing the historical data. Each data.frame must contain a column named "time" with the survival times and a column named "status" with the censoring status.
distributions	vector of parametric families for which prior distributions ought to be calibrated
prior_mu	prior distribution for the the meta-analytic mean parameter
prior_tau	prior distribution for the the meta-analytic heterogeneity parameter
...	additional parameters to be passed to the meta-analytic function. See BayesTools::JAGS_fit for more details.

Value

returns a list of prior distribution for the intercepts and auxiliary parameters.

---

calibrate_quartiles *Calibrate prior distributions based on quartiles*

Description

Calibrates prior distributions for parametric survival analysis based on median survival and interquartile range. Returns a list of prior distribution for the intercepts and auxiliary parameters.

Usage

```
calibrate_quartiles(
  median_t,
  iq_range_t,
  prior_sd = 0.5,
 distributions = c("exp-aft", "weibull-aft", "lnorm-aft", "llogis-aft", "gamma-aft"),
  verbose = FALSE,
  search_bounds1 = c(-100, 100),
  search_bounds2 = c(0 + 0.01, 100)
)
```

Arguments

median_t	median survival
iq_range_t	interquartile range of the survival
prior_sd	pre-specified standard deviation of the prior distributions (either a single value that is used for both the intercept and auxiliary parameter or a vector where the first value corresponds to the sd for the prior distribution on the intercept and the second value to the sd for the prior distribution on the auxiliary parameter)
distributions	vector of parametric families for which prior distributions ought to be calibrated
verbose	whether debug information be printed
search_bounds1	search boundaries for the intercept parameter
search_bounds2	search boundaries for the auxiliary parameter

Value

returns a list of prior distribution for the intercepts and auxiliary parameters.

Examples

```
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
```

---

check_RoBSA *Check fitted RoBSA object for errors and warnings*

Description

Checks fitted RoBSA object for warnings and errors and prints them to the console.

Usage

```
check_RoBSA(fit)
```

Arguments

fit	a fitted RoBSA object.

Value

check_RoBSA returns a vector of error and warning messages.

---

check_setup *Prints summary of "RoBSA" corresponding to the input*

Description

check_setup prints summary of "RoBSA" ensemble corresponding to the specified formula, data, and priors. This function is useful for checking the ensemble configuration prior to fitting all models.

Usage

```
check_setup(
  formula,
  data,
  priors = NULL,
  test_predictors = NULL,
 distributions = c("exp-aft", "weibull-aft", "lnorm-aft", "llogis-aft", "gamma-aft"),
  distributions_weights = rep(1, length(distributions)),
  prior_beta_null = get_default_prior_beta_null(),
  prior_beta_alt = get_default_prior_beta_alt(),
  prior_factor_null = get_default_prior_factor_null(),
  prior_factor_alt = get_default_prior_factor_alt(),
  prior_intercept = get_default_prior_intercept(),
  prior_aux = get_default_prior_aux(),
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = FALSE,
  rescale_data = FALSE,
  models = FALSE,
  ...
)
```

Arguments

formula	formula for the survival model
data	data frame containing the data
priors	names list of prior distributions for each predictor. It allows users to specify both the null and alternative hypothesis prior distributions by assigning a named list (with "null" and "alt" object) to the predictor
test_predictors	vector of predictor names to be tested with Bayesian model-averaged testing. Defaults to NULL, no parameters are tested.
distributions	distributions of parametric survival models
distributions_weights	prior odds for the competing distributions
prior_beta_null	default prior distribution for the null hypotheses of continuous predictors
prior_beta_alt	default prior distribution for the alternative hypotheses of continuous predictors

prior_factor_null
: default prior distribution for the null hypotheses of categorical predictors

prior_factor_alt
: default prior distribution for the alternative hypotheses of categorical predictors

prior_intercept
: named list containing prior distribution for the intercepts (with names corresponding to the distributions)

prior_aux
: named list containing prior distribution for the auxiliary parameters (with names corresponding to the distributions)

chains
: a number of chains of the MCMC algorithm.

sample
: a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.

burnin
: a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.

adapt
: a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.

thin
: a thinning of the chains of the MCMC algorithm. Defaults to `1`.

parallel
: whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.

autofit
: whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.

autofit_control
: allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.

convergence_checks
: automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.

save
: whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.

seed
: a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.

silent
: do not print the results.

rescale_data
: whether continuous predictors should be rescaled prior to estimating the model. Defaults to `FALSE`.

models
: should the models' details be printed.

...
: additional arguments.

Value

`check_setup` invisibly returns list of summary tables.

See Also

`RoBSA()`

`contr.BayesTools` *BayesTools Contrast Matrices*

Description

BayesTools provides several contrast matrix functions for Bayesian factor analysis. These functions create different types of contrast matrices that can be used with factor variables in Bayesian models.

Usage

```
contr.orthonormal(n, contrasts = TRUE)

contr.meandif(n, contrasts = TRUE)

contr.independent(n, contrasts = TRUE)
```

Arguments

`n`	a vector of levels for a factor, or the number of levels
`contrasts`	logical indicating whether contrasts should be computed

Details

The package includes the following contrast functions:

`contr.orthonormal` Return a matrix of orthonormal contrasts. Code is based on `stanova::contr.bayes` and corresponding to description by Rouder et al. (2012). Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.meandif` Return a matrix of mean difference contrasts. This is an adjustment to the `contr.orthonormal` that ascertains that the prior distributions on difference between the grand mean and factor level are identical independent of the number of factor levels (which does not hold for the orthonormal contrast). Furthermore, the contrast is re-scaled so the specified prior distribution exactly corresponds to the prior distribution on difference between each factor level and the grand mean – this is approximately twice the scale of `contr.orthonormal`. Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.independent` Return a matrix of independent contrasts – a level for each term. Returns a matrix with n rows and k columns, with k = n if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

References

Rouder JN, Morey RD, Speckman PL, Province JM (2012). "Default Bayes factors for ANOVA designs." *Journal of Mathematical Psychology*, **56**(5), 356–374. doi:10.1016/j.jmp.2012.08.001.

Examples

```
# Orthonormal contrasts
contr.orthonormal(c(1, 2))
contr.orthonormal(c(1, 2, 3))

# Mean difference contrasts
contr.meandif(c(1, 2))
contr.meandif(c(1, 2, 3))

# Independent contrasts
contr.independent(c(1, 2))
contr.independent(c(1, 2, 3))
```

---

default_prior *Default prior distributions*

Description

Functions for setting default prior distributions. Note that these default prior distributions might (and probably won't) apply to your specific data scenario.

Usage

```
get_default_prior_beta_null()

get_default_prior_beta_alt()

get_default_prior_factor_null()

get_default_prior_factor_alt()

get_default_prior_intercept()

get_default_prior_aux()
```

Value

`get_default_prior_beta_null` and `get_default_prior_beta_alt` return a prior distribution and `get_default_prior_intercept` and `get_default_prior_aux` return a list of prior distributions.

diagnostics *Visualizes MCMC diagnostics for a fitted RoBSA object*

Description

diagnostics creates visual checks of individual models convergence. Numerical overview of individual models can be obtained by summary(object, type = "diagnostics"), or even more detailed information by summary(object, type = "individual").

Usage

```
diagnostics(
  fit,
  parameter = NULL,
  type,
  plot_type = "base",
  show_models = NULL,
  lags = 30,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_autocorrelation(
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  lags = 30,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_trace(
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_density(
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  title = is.null(show_models) | length(show_models) > 1,
```

```
  ...
)
```

Arguments

`fit`	a fitted RoBSA object
`parameter`	a parameter to be plotted.
`type`	type of MCMC diagnostic to be plotted. Options are `"trace"` for the chains' trace plots, `"autocorrelation"` for autocorrelation of the chains, and `"densities"` for the overlaying densities of the individual chains. Can be abbreviated to first letters.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`show_models`	MCMC diagnostics of which models should be plotted. Defaults to `NULL` which plots MCMC diagnostics for a specified parameter for every model that is part of the ensemble.
`lags`	number of lags to be shown for `type = "autocorrelation"`. Defaults to `30`.
`title`	whether the model number should be displayed in title. Defaults to `TRUE` when more than one model is selected.
`...`	additional arguments to be passed to the plotting functions.

Value

`diagnostics` returns either `NULL` if `plot_type = "base"` or an object/list of objects (depending on the number of parameters to be plotted) of class 'ggplot2' if `plot_type = "ggplot2"`.

See Also

`RoBSA()`, `summary.RoBSA()`

---

exp-aft *Exponential AFT parametric family.*

Description

(log) density, hazard, and survival functions for AFT exponential parametric family.

Usage

```
exp_aft_log_density(t, eta)

exp_aft_log_hazard(t, eta)

exp_aft_log_survival(t, eta)
```

```
exp_aft_density(t, eta)

exp_aft_hazard(t, eta)

exp_aft_survival(t, eta)

exp_aft_mean(eta)

exp_aft_sd(eta)

exp_aft_r(n, eta)

exp_aft_q(p, eta)

exp_aft_p(q, eta)
```

Arguments

t	vector of survival times
eta	linear predictor
n	number of observations
p	vector of probabilities
q	vector of quantiles

Value

exp_aft_density, exp_aft_hazard, and exp_aft_survival return the density, hazard, and survival of the specified survival distribution. The exp_aft_log_density, exp_aft_log_hazard, exp_aft_log_survival return log of the corresponding qualities. exp_aft_mean and exp_aft_sd return the mean and standard deviation of the specified survival distribution. exp_aft_r, exp_aft_q, and exp_aft_p return a random generation, quantiles, and cumulative probabilities of the specified survival distribution.

---

extract_flexsurv *Extract parameter estimates from* flexsurv *object*

Description

extract_flexsurv extracts estimates from a flexsurv object in and transform them to match the RoBSA output.

Usage

```
extract_flexsurv(fit)
```

Arguments

fit	an object fitted with the flexsurv::flexsurvreg function

Value

extract_flexsurv return list of estimates lists for each parameter.

---

gamma-aft *Gamma AFT parametric family.*

Description

(log) density, hazard, and survival functions for AFT gamma parametric family.

Usage

```
gamma_aft_log_density(t, eta, shape)

gamma_aft_log_hazard(t, eta, shape)

gamma_aft_log_survival(t, eta, shape)

gamma_aft_density(t, eta, shape)

gamma_aft_hazard(t, eta, shape)

gamma_aft_survival(t, eta, shape)

gamma_aft_mean(eta, shape)

gamma_aft_sd(eta, shape)

gamma_aft_r(n, eta, shape)

gamma_aft_q(p, eta, shape)

gamma_aft_p(q, eta, shape)
```

Arguments

t	vector of survival times
eta	linear predictor
shape	auxiliary parameter
n	number of observations
p	vector of probabilities
q	vector of quantiles

Value

gamma_aft_density, gamma_aft_hazard, and gamma_aft_survival return the density, hazard, and survival of the specified survival distribution. The gamma_aft_log_density, gamma_aft_log_hazard, gamma_aft_log_survival return log of the corresponding qualities. gamma_aft_mean and gamma_aft_sd return the mean and standard deviation of the specified survival distribution. gamma_aft_r, gamma_aft_q, and gamma_aft_p return a random generation, quantiles, and cumulative probabilities of the specified survival distribution.

---

is.RoBSA *Reports whether x is a RoBSA object*

Description

Reports whether x is a RoBSA object

Usage

```
is.RoBSA(x)
```

Arguments

x	an object to test

Value

is.RoBSA returns a boolean.

---

llogis-aft *Log-logistic AFT parametric family.*

Description

(log) density, hazard, and survival functions for AFT log-logistic parametric family.

Usage

```
llogis_aft_log_density(t, eta, shape)

llogis_aft_log_hazard(t, eta, shape)

llogis_aft_log_survival(t, eta, shape)

llogis_aft_density(t, eta, shape)

llogis_aft_hazard(t, eta, shape)
```

```
llogis_aft_survival(t, eta, shape)

llogis_aft_mean(eta, shape)

llogis_aft_sd(eta, shape)

llogis_aft_r(n, eta, shape)

llogis_aft_q(p, eta, shape)

llogis_aft_p(q, eta, shape)
```

Arguments

t	vector of survival times
eta	linear predictor
shape	auxiliary parameter
n	number of observations
p	vector of probabilities
q	vector of quantiles

Value

`llogis_aft_density`, `llogis_aft_hazard`, and `llogis_aft_survival` return the density, hazard, and survival of the specified survival distribution. The `llogis_aft_log_density`, `llogis_aft_log_hazard`, `llogis_aft_log_survival` return log of the corresponding qualities. `llogis_aft_mean` and `llogis_aft_sd` return the mean and standard deviation of the specified survival distribution. `llogis_aft_r`, `llogis_aft_q`, and `llogis_aft_p` return a random generation, quantiles, and cumulative probabilities of the specified survival distribution.

---

lnorm-aft *Log-normal AFT parametric family.*

Description

(log) density, hazard, and survival functions for AFT log-normal parametric family.

Usage

```
lnorm_aft_log_density(t, eta, sd)

lnorm_aft_log_hazard(t, eta, sd)

lnorm_aft_log_survival(t, eta, sd)
```

```
lnorm_aft_density(t, eta, sd)

lnorm_aft_hazard(t, eta, sd)

lnorm_aft_survival(t, eta, sd)

lnorm_aft_mean(eta, sd)

lnorm_aft_sd(eta, sd)

lnorm_aft_r(n, eta, sd)

lnorm_aft_q(p, eta, sd)

lnorm_aft_p(q, eta, sd)
```

Arguments

t	vector of survival times
eta	linear predictor
sd	auxiliary parameter
n	number of observations
p	vector of probabilities
q	vector of quantiles

Value

lnorm_aft_density, lnorm_aft_hazard, and lnorm_aft_survival return the density, hazard, and survival of the specified survival distribution. The lnorm_aft_log_density, lnorm_aft_log_hazard, lnorm_aft_log_survival return log of the corresponding qualities. lnorm_aft_mean and lnorm_aft_sd return the mean and standard deviation of the specified survival distribution. lnorm_aft_r, lnorm_aft_q, and lnorm_aft_p return a random generation, quantiles, and cumulative probabilities of the specified survival distribution.

---

plot.RoBSA *Plots a fitted RoBSA object*

Description

plot.RoBSA allows to visualize posterior distribution of different "RoBSA" object parameters. See plot_survival for plotting the survival ways. See type for the different model types.

Usage

```
## S3 method for class 'RoBSA'
plot(
  x,
  parameter = NULL,
  conditional = FALSE,
  plot_type = "base",
  prior = FALSE,
  dots_prior = NULL,
  ...
)
```

Arguments

x	a fitted RoBSA object
parameter	a name of parameter to be plotted. Defaults to the first regression parameter if left unspecified. Use "intercept" and "aux" to plot the intercepts and auxiliary parameters of each distribution family.
conditional	whether conditional estimates should be plotted. Defaults to FALSE which plots the model-averaged estimates.
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
prior	whether prior distribution should be added to figure. Defaults to FALSE.
dots_prior	list of additional graphical arguments to be passed to the plotting function of the prior distribution. Supported arguments are lwd, lty, col, and col.fill, to adjust the line thickness, line type, line color, and fill color of the prior distribution respectively.
...	list of additional graphical arguments to be passed to the plotting function. Supported arguments are lwd, lty, col, col.fill, xlab, ylab, main, xlim, ylim to adjust the line thickness, line type, line color, fill color, x-label, y-label, title, x-axis range, and y-axis range respectively.

Value

plot.RoBSA returns either NULL if plot_type = "base" or an object object of class 'ggplot2' if plot_type = "ggplot2".

See Also

RoBSA()

Examples

```
## Not run:
# (execution of the example takes several minutes)
# example from the README (more details and explanation therein)
data(cancer, package = "survival")
```

```
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
df <- data.frame(
  time         = veteran$time / 12,
  status       = veteran$status,
 treatment   = factor(ifelse(veteran$trt == 1, "standard", "new"), levels = c("standard", "new")),
  karno_scaled = veteran$karno / 100
)
RoBSA.options(check_scaling = FALSE)
fit <- RoBSA(
  Surv(time, status) ~ treatment + karno_scaled,
  data   = df,
  priors = list(
    treatment    = prior_factor("normal", parameters = list(mean = 0.30, sd = 0.15),
                                truncation = list(0, Inf), contrast = "treatment"),
    karno_scaled = prior("normal", parameters = list(mean = 0, sd = 1))
  ),
  test_predictors = "treatment",
  prior_intercept = priors[["intercept"]],
  prior_aux       = priors[["aux"]],
  parallel = TRUE, seed = 1
)

# plot posterior distribution of the treatment effect
plot(fit, parameter = "treatment")


## End(Not run)
```

`plot_models` *Models plot for a RoBSA object*

Description

`plot_models` plots individual models' estimates for a `"RoBSA"` object.

Usage

```
plot_models(
  x,
  parameter = NULL,
  conditional = FALSE,
  plot_type = "base",
  order = "decreasing",
  order_by = "model",
  ...
)
```

Arguments

x	a fitted RoBSA object
parameter	a name of parameter to be plotted. Defaults to the first regression parameter if left unspecified.
conditional	whether conditional estimates should be plotted. Defaults to FALSE which plots the model-averaged estimates.
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
order	how the models should be ordered. Defaults to "decreasing" which orders them in decreasing order in accordance to order_by argument. The alternative is "increasing".
order_by	what feature should be use to order the models. Defaults to "model" which orders the models according to their number. The alternatives are "estimate" (for the effect size estimates), "probability" (for the posterior model probability), and "BF" (for the inclusion Bayes factor).
...	list of additional graphical arguments to be passed to the plotting function. Supported arguments are lwd, lty, col, col.fill, xlab, ylab, main, xlim, ylim to adjust the line thickness, line type, line color, fill color, x-label, y-label, title, x-axis range, and y-axis range respectively.

Value

plot_models returns either NULL if plot_type = "base" or an object object of class 'ggplot2' if plot_type = "ggplot2".

Examples

```
## Not run:
# (execution of the example takes several minutes)
# example from the README (more details and explanation therein)
data(cancer, package = "survival")
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
df <- data.frame(
  time         = veteran$time / 12,
  status       = veteran$status,
 treatment   = factor(ifelse(veteran$trt == 1, "standard", "new"), levels = c("standard", "new")),
  karno_scaled = veteran$karno / 100
)
RoBSA.options(check_scaling = FALSE)
fit <- RoBSA(
  Surv(time, status) ~ treatment + karno_scaled,
  data   = df,
  priors = list(
    treatment    = prior_factor("normal", parameters = list(mean = 0.30, sd = 0.15),
                                truncation = list(0, Inf), contrast = "treatment"),
    karno_scaled = prior("normal", parameters = list(mean = 0, sd = 1))
  ),
  test_predictors = "treatment",
```

```
  prior_intercept = priors[["intercept"]],
  prior_aux       = priors[["aux"]],
  parallel = TRUE, seed = 1
)

# plot posterior distribution of the treatment effect from each model
plot_models(fit, parameter = "treatment")


## End(Not run)
```

---

plot_prediction *Survival plots for a RoBSA object*

Description

Survival plots for a RoBSA object

Usage

```
plot_prediction(
  x,
  type = "survival",
  time_range = NULL,
  new_data = NULL,
  predictor = NULL,
  covariates_data = NULL,
  conditional = FALSE,
  plot_type = "base",
  samples = 10000,
  ...
)

plot_survival(
  x,
  time_range = NULL,
  new_data = NULL,
  predictor = NULL,
  covariates_data = NULL,
  conditional = FALSE,
  plot_type = "base",
  samples = 10000,
  ...
)

plot_hazard(
```

```
  x,
  time_range = NULL,
  new_data = NULL,
  predictor = NULL,
  covariates_data = NULL,
  conditional = FALSE,
  plot_type = "base",
  samples = 10000,
  ...
)

plot_density(
  x,
  time_range = NULL,
  new_data = NULL,
  predictor = NULL,
  covariates_data = NULL,
  conditional = FALSE,
  plot_type = "base",
  samples = 10000,
  ...
)
```

Arguments

x	a fitted RoBSA object.
type	what type of prediction should be created
time_range	a numeric of length two specifying the range for the survival prediction. Defaults to NULL which uses the range of observed times.
new_data	a data.frame containing fully specified predictors for which predictions should be made
predictor	an alternative input to new_data that automatically generates predictions for each level of the predictor across all either across levels of covariates specified by covariates_data or at the default values of other predictors
covariates_data	a supplementary input to predictor that specifies levels of covariates for which predictions should be made
conditional	whether only models assuming presence of the specified predictor should be used
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
samples	number of posterior samples to be evaluated
...	additional arguments.

Value

returns either NULL if plot_type = "base" or an object object of class 'ggplot2' if plot_type = "ggplot2".

Examples

```
## Not run:
# (execution of the example takes several minutes)
# example from the README (more details and explanation therein)
data(cancer, package = "survival")
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
df <- data.frame(
  time         = veteran$time / 12,
  status       = veteran$status,
 treatment   = factor(ifelse(veteran$trt == 1, "standard", "new"), levels = c("standard", "new")),
  karno_scaled = veteran$karno / 100
)
RoBSA.options(check_scaling = FALSE)
fit <- RoBSA(
  Surv(time, status) ~ treatment + karno_scaled,
  data   = df,
  priors = list(
    treatment    = prior_factor("normal", parameters = list(mean = 0.30, sd = 0.15),
                                truncation = list(0, Inf), contrast = "treatment"),
    karno_scaled = prior("normal", parameters = list(mean = 0, sd = 1))
  ),
  test_predictors = "treatment",
  prior_intercept = priors[["intercept"]],
  prior_aux       = priors[["aux"]],
  parallel = TRUE, seed = 1
)

# plot survival for each level the treatment
plot_survival(fit, parameter = "treatment")

# plot hazard for each level the treatment
plot_hazard(fit, parameter = "treatment")

# plot density for each level the treatment
plot_density(fit, parameter = "treatment")

## End(Not run)
```

`predict.RoBSA` *Predict method for RoBSA objects.*

Description

Predicts survival/hazard/density/mean/sd for a given RoBSA object. Either predicts values for each row of a fully specified `new_data` data.frame, or for all levels of a given `predictor` at the mean of continuous covariate values and default factor levels or covariate values specified as `covariates_data` data.frame.

Usage

```
## S3 method for class 'RoBSA'
predict(
  object,
  time = NULL,
  new_data = NULL,
  predictor = NULL,
  covariates_data = NULL,
  type = c("survival", "hazard", "density", "mean", "sd"),
  summarize = TRUE,
  averaged = TRUE,
  conditional = FALSE,
  samples = 10000,
  ...
)
```

Arguments

`object`	a fitted RoBSA object
`time`	a vector of time values at which the survival/hazard/density will be predicted (for each passed data point)
`new_data`	a data.frame containing fully specified predictors for which predictions should be made
`predictor`	an alternative input to `new_data` that automatically generates predictions for each level of the predictor across all either across levels of covariates specified by `covariates_data` or at the default values of other predictors
`covariates_data`	a supplementary input to `predictor` that specifies levels of covariates for which predictions should be made
`type`	what type of prediction should be created
`summarize`	whether the predictions should be aggregated as mean and sd. Otherwise, prediction for for posterior samples is returned.
`averaged`	whether predictions should be combined with Bayesian model-averaging or whether predictions for each individual model should be returned.
`conditional`	whether only models assuming presence of the specified `predictor` should be used
`samples`	number of posterior samples to be evaluated
`...`	additional arguments (unused)

Value

a list with predictions (or a list of lists in case that predictions for each individual model are requested `averaged = FALSE`)

print.RoBSA *Prints a fitted RoBSA object*

Description

Prints a fitted RoBSA object

Usage

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

Arguments

x	a fitted RoBSA object.
...	additional arguments.

Value

print.RoBSA invisibly returns the print statement.

See Also

RoBSA()

print.summary.RoBSA *Prints summary object for RoBSA method*

Description

Prints summary object for RoBSA method

Usage

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

Arguments

x	a summary of a RoBSA object
...	additional arguments

Value

print.summary.RoBSA invisibly returns the print statement.

See Also

RoBSA()

---

prior *Creates a prior distribution*

Description

prior creates a prior distribution. The prior can be visualized by the plot function.

Usage

```
prior(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1
)
```

Arguments

distribution	name of the prior distribution. The possible options are "point" for a point density characterized by a location parameter. "normal" for a normal distribution characterized by a mean and sd parameters. "lognormal" for a lognormal distribution characterized by a meanlog and sdlog parameters. "cauchy" for a Cauchy distribution characterized by a location and scale parameters. Internally converted into a generalized t-distribution with df = 1. "t" for a generalized t-distribution characterized by a location, scale, and df parameters. "gamma" for a gamma distribution characterized by either shape and rate, or shape and scale parameters. The later is internally converted to the shape and rate parametrization "invgamma" for an inverse-gamma distribution characterized by a shape and scale parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. "beta" for a beta distribution characterized by an alpha and beta parameters. "exp" for an exponential distribution characterized by either rate or scale parameter. The later is internally converted to rate. "uniform" for a uniform distribution defined on a range from a to b
parameters	list of appropriate parameters for a given distribution.
truncation	list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.

prior_weights prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

prior and prior_none return an object of class 'prior'. A named list containing the distribution name, parameters, and prior weights.

See Also

plot.prior(), Normal, Lognormal, Cauchy, Beta, Exponential, LocationScaleT, InvGamma.

Examples

```
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# create a half-normal standard normal prior distribution
p2 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1),
truncation = list(lower = 0, upper = Inf))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)
```

---

prior_factor *Creates a prior distribution for factors*

Description

prior_factor creates a prior distribution for fitting models with factor predictors. (Note that results across different operating systems might vary due to differences in JAGS numerical precision.)

Usage

```
prior_factor(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1,
  contrast = "orthonormal"
)
```

Arguments

distribution	name of the prior distribution. The possible options are "point" for a point density characterized by a location parameter. "normal" for a normal distribution characterized by a mean and sd parameters. "lognormal" for a lognormal distribution characterized by a meanlog and sdlog parameters. "cauchy" for a Cauchy distribution characterized by a location and scale parameters. Internally converted into a generalized t-distribution with df = 1. "t" for a generalized t-distribution characterized by a location, scale, and df parameters. "gamma" for a gamma distribution characterized by either shape and rate, or shape and scale parameters. The later is internally converted to the shape and rate parametrization "invgamma" for an inverse-gamma distribution characterized by a shape and scale parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. "beta" for a beta distribution characterized by an alpha and beta parameters. "exp" for an exponential distribution characterized by either rate or scale parameter. The later is internally converted to rate. "uniform" for a uniform distribution defined on a range from a to b
parameters	list of appropriate parameters for a given distribution.
truncation	list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.
prior_weights	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.
contrast	type of contrast for the prior distribution. The possible options are "meandif" for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. In contrast to "orthonormal", the marginal distributions are identical regardless of the number of factor levels and the specified prior distribution corresponds to the difference from grand mean for each factor level. Only supports distribution = "mnormal" and distribution = "mt" which generates the corresponding multivariate normal/t distributions. "orthonormal" for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. Only supports distribution = "mnormal" and distribution = "mt" which generates the corresponding multivariate normal/t distributions. "treatment" for contrasts using the first level as a comparison group and setting equal prior distribution on differences between the individual factor levels and the comparison level.

"independent" for contrasts specifying dependent prior distribution for each factor level (note that this leads to an overparameterized model if the intercept is included).

Value

return an object of class 'prior'.

See Also

prior()

Examples

```
# create an orthonormal prior distribution
p1 <- prior_factor(distribution = "mnormal", contrast = "orthonormal",
                   parameters = list(mean = 0, sd = 1))
```

---

prior_informed *Creates an informed prior distribution based on research*

Description

prior_informed creates an informed prior distribution based on past research. The prior can be visualized by the plot function.

Usage

```
prior_informed(name, parameter = NULL, type = "smd")
```

Arguments

name	name of the prior distribution. There are many options based on prior psychological or medical research. For psychology, the possible options are
	"van Erp" for an informed prior distribution for the heterogeneity parameter tau of meta-analytic effect size estimates based on standardized mean differences (van Erp et al. 2017),
	"Oosterwijk" for an informed prior distribution for the effect sizes expected in social psychology based on prior elicitation with dr. Oosterwijk (Gronau et al. 2017).
	For medicine, the possible options are based on Bartoš et al. (2021) who developed empirical prior distributions for the effect size and heterogeneity parameters of the continuous standardized outcomes based on the Cochrane database of systematic reviews. Use "Cochrane" for a prior distribution based on the whole database or call print(prior_informed_medicine_names) to inspect the names of all 46 subfields and set the appropriate parameter and type.

parameter	parameter name describing what prior distribution is supposed to be produced in cases where the name corresponds to multiple prior distributions. Relevant only for the empirical medical prior distributions.
type	prior type describing what prior distribution is supposed to be produced in cases where the name and parameter correspond to multiple prior distributions. Relevant only for the empirical medical prior distributions.

Details

Further details can be found in van Erp et al. (2017), Gronau et al. (2017), and Bartoš et al. (2021).

Value

prior_informed returns an object of class 'prior'.

References

Bartoš F, Gronau QF, Timmers B, Otte WM, Ly A, Wagenmakers E (2021). "Bayesian model-averaged meta-analysis in medicine." *Statistics in Medicine*, **40**(30), 6743–6761. doi:10.1002/sim.9170.

Gronau QF, Van Erp S, Heck DW, Cesario J, Jonas KJ, Wagenmakers E (2017). "A Bayesian model-averaged meta-analysis of the power pose effect with informed and default priors: The case of felt power." *Comprehensive Results in Social Psychology*, **2**(1), 123–138. doi:10.1080/23743603.2017.1326760.

van Erp S, Verhagen J, Grasman RP, Wagenmakers E (2017). "Estimates of between-study heterogeneity for 705 meta-analyses reported in Psychological Bulletin from 1990–2013." *Journal of Open Psychology Data*, **5**(1). doi:10.5334/jopd.33.

See Also

prior(), prior_informed_medicine_names

Examples

```
# prior distribution representing expected effect sizes in social psychology
# based on prior elicitation with dr. Oosterwijk
p1 <- prior_informed("Oosterwijk")

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)


# empirical prior distribution for the standardized mean differences from the oral health
# medical subfield based on meta-analytic effect size estimates from the
# Cochrane database of systematic reviews
p2 <- prior_informed("Oral Health", parameter = "effect", type = "smd")
print(p2)
```

prior_informed_medicine_names

Names of medical subfields from the Cochrane database of systematic reviews

Description

Contain names identifying the individual subfields from the Cochrane database of systematic reviews. The individual elements correspond to valid name arguments for the `prior_informed()` function.

Usage

```
prior_informed_medicine_names
```

Format

An object of class `character` of length 47.

prior_none

Creates a prior distribution

Description

`prior` creates a prior distribution. The prior can be visualized by the `plot` function.

Usage

```
prior_none(prior_weights = 1)
```

Arguments

`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

`prior` and `prior_none` return an object of class 'prior'. A named list containing the distribution name, parameters, and prior weights.

See Also

`plot.prior()`, Normal, Lognormal, Cauchy, Beta, Exponential, LocationScaleT, InvGamma.

Examples

```
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# create a half-normal standard normal prior distribution
p2 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1),
truncation = list(lower = 0, upper = Inf))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)
```

RoBSA *Fit Robust Bayesian Survival Analysis*

Description

RoBSA is used to estimate a robust Bayesian survival analysis. The interface allows a complete customization of the ensemble with different prior distributions for the null and alternative hypothesis of each parameter. (See README for an example.)

Usage

```
RoBSA(
  formula,
  data,
  priors = NULL,
  test_predictors = NULL,
 distributions = c("exp-aft", "weibull-aft", "lnorm-aft", "llogis-aft", "gamma-aft"),
  distributions_weights = rep(1, length(distributions)),
  prior_beta_null = get_default_prior_beta_null(),
  prior_beta_alt = get_default_prior_beta_alt(),
  prior_factor_null = get_default_prior_factor_null(),
  prior_factor_alt = get_default_prior_factor_alt(),
  prior_intercept = get_default_prior_intercept(),
  prior_aux = get_default_prior_aux(),
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
```

```
  silent = TRUE,
  rescale_data = FALSE,
  ...
)
```

Arguments

formula	formula for the survival model
data	data frame containing the data
priors	names list of prior distributions for each predictor. It allows users to specify both the null and alternative hypothesis prior distributions by assigning a named list (with "null" and "alt" object) to the predictor
test_predictors	vector of predictor names to be tested with Bayesian model-averaged testing. Defaults to NULL, no parameters are tested.
distributions	distributions of parametric survival models
distributions_weights	prior odds for the competing distributions
prior_beta_null	default prior distribution for the null hypotheses of continuous predictors
prior_beta_alt	default prior distribution for the alternative hypotheses of continuous predictors
prior_factor_null	default prior distribution for the null hypotheses of categorical predictors
prior_factor_alt	default prior distribution for the alternative hypotheses of categorical predictors
prior_intercept	named list containing prior distribution for the intercepts (with names corresponding to the distributions)
prior_aux	named list containing prior distribution for the auxiliary parameters (with names corresponding to the distributions)
chains	a number of chains of the MCMC algorithm.
sample	a number of sampling iterations of the MCMC algorithm. Defaults to 5000.
burnin	a number of burnin iterations of the MCMC algorithm. Defaults to 2000.
adapt	a number of adaptation iterations of the MCMC algorithm. Defaults to 500.
thin	a thinning of the chains of the MCMC algorithm. Defaults to 1.
parallel	whether the individual models should be fitted in parallel. Defaults to FALSE. The implementation is not completely stable and might cause a connection error.
autofit	whether the model should be fitted until the convergence criteria (specified in autofit_control) are satisfied. Defaults to TRUE.
autofit_control	allows to pass autofit control settings with the set_autofit_control() function. See ?set_autofit_control for options and default settings.

convergence_checks
: automatic convergence checks to assess the fitted models, passed with set_convergence_checks() function. See ?set_convergence_checks for options and default settings.

save
: whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to "all" which does not remove anything. Set to "min" to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.

seed
: a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to NULL - no seed is set.

silent
: whether all print messages regarding the fitting process should be suppressed. Defaults to TRUE. Note that parallel = TRUE also suppresses all messages.

rescale_data
: whether continuous predictors should be rescaled prior to estimating the model. Defaults to FALSE.

...
: additional arguments.

Value

RoBSA returns an object of class 'RoBSA'.

Examples

```
## Not run:
# (execution of the example takes several minutes)
# example from the README (more details and explanation therein)
data(cancer, package = "survival")
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
df <- data.frame(
  time         = veteran$time / 12,
  status       = veteran$status,
 treatment   = factor(ifelse(veteran$trt == 1, "standard", "new"), levels = c("standard", "new")),
  karno_scaled = veteran$karno / 100
)
RoBSA.options(check_scaling = FALSE)
fit <- RoBSA(
  Surv(time, status) ~ treatment + karno_scaled,
  data   = df,
  priors = list(
    treatment    = prior_factor("normal", parameters = list(mean = 0.30, sd = 0.15),
                                truncation = list(0, Inf), contrast = "treatment"),
    karno_scaled = prior("normal", parameters = list(mean = 0, sd = 1))
  ),
  test_predictors = "treatment",
  prior_intercept = priors[["intercept"]],
  prior_aux       = priors[["aux"]],
  parallel = TRUE, seed = 1
)
summary(fit)


## End(Not run)
```

RoBSA_control *Control MCMC fitting process*

Description

Controls settings for the autofit process of the MCMC JAGS sampler (specifies termination criteria), and values for the convergence checks.

Usage

```
set_autofit_control(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  max_time = list(time = 60, unit = "mins"),
  sample_extend = 1000,
  restarts = 10,
  max_extend = 10,
  check_indicators = FALSE
)

set_convergence_checks(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  check_indicators = FALSE,
  remove_failed = FALSE,
  balance_probability = TRUE
)
```

Arguments

max_Rhat	maximum value of the R-hat diagnostic. Defaults to 1.05.
min_ESS	minimum estimated sample size. Defaults to 500.
max_error	maximum value of the MCMC error. Defaults to NULL. Be aware that PEESE publication bias adjustment can have estimates on different scale than the rest of the output, resulting in relatively large max MCMC error.
max_SD_error	maximum value of the proportion of MCMC error of the estimated SD of the parameter. Defaults to NULL.
max_time	list with the time and unit specifying the maximum autofitting process per model. Passed to difftime function (possible units are "secs", "mins", "hours", "days", "weeks", "years"). Defaults to list(time = 60, unit = "mins").
sample_extend	number of samples to extend the fitting process if the criteria are not satisfied. Defaults to 1000.

`restarts` number of fitting restarts attempted when the initial JAGS model fitting fails. Defaults to `10`.

`max_extend` maximum number of autofit extensions per model. Defaults to `10`.

`check_indicators` whether convergence checks should include latent prior inclusion indicators. Defaults to `FALSE`.

`remove_failed` whether models not satisfying the convergence checks should be removed from the inference. Defaults to `FALSE` - only a warning is raised.

`balance_probability` whether prior model probability should be balanced across the combinations of models with the same H0/H1 for effect / heterogeneity / bias in the case of non-convergence. Defaults to `TRUE`.

Value

`set_autofit_control` returns a list of autofit control settings and `set_convergence_checks` returns a list of convergence checks settings.

See Also

RoBSA, update.RoBSA

`RoBSA_options` *Options for the RoBSA package*

Description

A placeholder object and functions for the RoBSA package. (adapted from the runjags R package).

Usage

```
RoBSA.options(...)

RoBSA.get_option(name)
```

Arguments

`...` named option(s) to change - for a list of available options, see details below.

`name` the name of the option to get the current value of - for a list of available options, see details below.

Value

The current value of all available RoBSA options (after applying any changes specified) is returned invisibly as a named list.

summary.RoBSA *Summarize fitted RoBSA object*

Description

summary.RoBSA creates a numerical summary of the RoBSA object.

Usage

```
## S3 method for class 'RoBSA'
summary(
  object,
  type = "ensemble",
  conditional = FALSE,
  exp = FALSE,
  parameters = FALSE,
  probs = c(0.025, 0.975),
  logBF = FALSE,
  BF01 = FALSE,
  transform_factors = TRUE,
  short_name = FALSE,
  remove_spike_0 = FALSE,
  ...
)
```

Arguments

object	a fitted RoBSA object.
type	whether to show the overall RoBSA results ("ensemble"), an overview of the individual models ("models"), or detailed summary for the individual models ("individual").
conditional	show the conditional estimates (assuming that the alternative is true). Defaults to FALSE. Only available for type == "conditional".
exp	whether exponents of the regression estimates should be also presented
parameters	character vector of parameters (or a named list with of character vectors for summary and diagnostics tables) specifying the parameters (and their grouping) for the summary table
probs	quantiles of the posterior samples to be displayed. Defaults to c(.025, .50, .975)
logBF	show log of the BFs. Defaults to FALSE.
BF01	show BF in support of the null hypotheses. Defaults to FALSE.
transform_factors	Whether factors with orthonormal prior distributions should be transformed to differences from the grand mean. Defaults to TRUE.

short_name	whether the prior distribution names should be shortened. Defaults to FALSE.
remove_spike_0	whether prior distributions equal to spike at 0 should be removed from the prior_list
...	additional arguments

Value

summary of a RoBSA object

summary.RoBSA returns a list of tables of class 'BayesTools_table'.

Note

See diagnostics() for visual convergence checks of the individual models.

See Also

RoBSA(), diagnostics(), check_RoBSA()

Examples

```
## Not run:
# (execution of the example takes several minutes)
# example from the README (more details and explanation therein)
data(cancer, package = "survival")
priors <- calibrate_quartiles(median_t = 5, iq_range_t = 10, prior_sd = 0.5)
df <- data.frame(
  time         = veteran$time / 12,
  status       = veteran$status,
 treatment   = factor(ifelse(veteran$trt == 1, "standard", "new"), levels = c("standard", "new")),
  karno_scaled = veteran$karno / 100
)
RoBSA.options(check_scaling = FALSE)
fit <- RoBSA(
  Surv(time, status) ~ treatment + karno_scaled,
  data   = df,
  priors = list(
    treatment    = prior_factor("normal", parameters = list(mean = 0.30, sd = 0.15),
                                truncation = list(0, Inf), contrast = "treatment"),
    karno_scaled = prior("normal", parameters = list(mean = 0, sd = 1))
  ),
  test_predictors = "treatment",
  prior_intercept = priors[["intercept"]],
  prior_aux       = priors[["aux"]],
  parallel = TRUE, seed = 1
)

# summary can provide many details about the model
summary(fit)

# note that the summary function contains additional arguments
# that allow to obtain a specific output, i.e, the conditional estimates
```

```
# (assuming that the non-null models are true) can be obtained
summary(fit, conditional = TRUE)

# overview of the models and their prior and posterior probability, marginal likelihood,
# and inclusion Bayes factor:
summary(fit, type = "models")

# and the model diagnostics overview, containing maximum R-hat and minimum ESS across parameters
# but see '?diagnostics' for diagnostics plots for individual model parameters
summary(fit, type = "diagnostics")

# summary of individual models and their parameters can be further obtained by
summary(fit, type = "individual")


## End(Not run)
```

`update.RoBSA` *Updates a fitted RoBSA object*

Description

`update.RoBSA` can be used to

1. add an additional model to an existing `"RoBSA"` object by specifying the distribution, and either null or alternative priors for each parameter and prior weight of the model,
2. change the prior weights of fitted models by specifying a vector `prior_weights` of the same length as the fitted models,
3. refitting models that failed to converge with updated settings of control parameters,
4. or changing the convergence criteria and recalculating the ensemble results by specifying new `control` argument and setting `refit_failed == FALSE`.

Usage

```
## S3 method for class 'RoBSA'
update(
  object,
  refit_failed = TRUE,
  formula = NULL,
  priors = NULL,
  test_predictors = "",
  distribution = NULL,
  model_weights = 1,
  prior_beta_null = get_default_prior_beta_null(),
  prior_beta_alt = get_default_prior_beta_alt(),
  prior_factor_null = get_default_prior_factor_null(),
  prior_factor_alt = get_default_prior_factor_alt(),
```

```
  prior_intercept = get_default_prior_intercept(),
  prior_aux = get_default_prior_aux(),
  chains = NULL,
  adapt = NULL,
  burnin = NULL,
  sample = NULL,
  thin = NULL,
  autofit = NULL,
  parallel = NULL,
  autofit_control = NULL,
  convergence_checks = NULL,
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

object	a fitted RoBSA object
refit_failed	whether failed models should be refitted. Relevant only if new priors or prior_weights are not supplied. Defaults to TRUE.
formula	formula for the survival model
priors	names list of prior distributions for each predictor. It allows users to specify both the null and alternative hypothesis prior distributions by assigning a named list (with "null" and "alt" object) to the predictor
test_predictors	vector of predictor names to be tested with Bayesian model-averaged testing. Defaults to NULL, no parameters are tested.
distribution	a distribution of the new model.
model_weights	either a single value specifying prior model weight of a newly specified model using priors argument, or a vector of the same length as already fitted models to update their prior weights.
prior_beta_null	default prior distribution for the null hypotheses of continuous predictors
prior_beta_alt	default prior distribution for the alternative hypotheses of continuous predictors
prior_factor_null	default prior distribution for the null hypotheses of categorical predictors
prior_factor_alt	default prior distribution for the alternative hypotheses of categorical predictors
prior_intercept	named list containing prior distribution for the intercepts (with names corresponding to the distributions)
prior_aux	named list containing prior distribution for the auxiliary parameters (with names corresponding to the distributions)

chains	a number of chains of the MCMC algorithm.
adapt	a number of adaptation iterations of the MCMC algorithm. Defaults to 500.
burnin	a number of burnin iterations of the MCMC algorithm. Defaults to 2000.
sample	a number of sampling iterations of the MCMC algorithm. Defaults to 5000.
thin	a thinning of the chains of the MCMC algorithm. Defaults to 1.
autofit	whether the model should be fitted until the convergence criteria (specified in autofit_control) are satisfied. Defaults to TRUE.
parallel	whether the individual models should be fitted in parallel. Defaults to FALSE. The implementation is not completely stable and might cause a connection error.
autofit_control	allows to pass autofit control settings with the set_autofit_control() function. See ?set_autofit_control for options and default settings.
convergence_checks	automatic convergence checks to assess the fitted models, passed with set_convergence_checks() function. See ?set_convergence_checks for options and default settings.
save	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to "all" which does not remove anything. Set to "min" to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
seed	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to NULL - no seed is set.
silent	whether all print messages regarding the fitting process should be suppressed. Defaults to TRUE. Note that parallel = TRUE also suppresses all messages.
...	additional arguments.

Details

See RoBSA() for more details.

Value

update.RoBSA returns an object of class 'RoBSA'.

See Also

RoBSA(), summary.RoBSA(), prior(), check_setup()

weibull-aft *Weibull AFT parametric family.*

Description

(log) density, hazard, and survival functions for AFT Weibull parametric family.

Usage

```
weibull_aft_log_density(t, eta, shape)

weibull_aft_log_hazard(t, eta, shape)

weibull_aft_log_survival(t, eta, shape)

weibull_aft_density(t, eta, shape)

weibull_aft_hazard(t, eta, shape)

weibull_aft_survival(t, eta, shape)

weibull_aft_mean(eta, shape)

weibull_aft_sd(eta, shape)

weibull_aft_r(n, eta, shape)

weibull_aft_q(p, eta, shape)

weibull_aft_p(q, eta, shape)
```

Arguments

t	vector of survival times
eta	linear predictor
shape	auxiliary parameter
n	number of observations
p	vector of probabilities
q	vector of quantiles

Value

`weibull_aft_density`, `weibull_aft_hazard`, and `weibull_aft_survival` return the density, hazard, and survival of the specified survival distribution. The `weibull_aft_log_density`, `weibull_aft_log_hazard`, `weibull_aft_log_survival` return log of the corresponding qualities. `weibull_aft_mean` and `weibull_aft_sd` return the mean and standard deviation of the specified survival distribution.

`weibull_aft_r`, `weibull_aft_q`, and `weibull_aft_p` return a random generation, quantiles, and cumulative probabilities of the specified survival distribution.

Index

∗ **datasets**
    prior_informed_medicine_names, 30
∗ **package**
    RoBSA-package, 3
_PACKAGE *(*RoBSA-package*)*, 3

BayesTools::JAGS_fit, *4*
Beta, *26*, *30*

calibrate_meta_analytic, 3
calibrate_quartiles, 4
Cauchy, *26*, *30*
check_RoBSA, 5
check_RoBSA(), *37*
check_setup, 5
check_setup(), *40*
contr.BayesTools, 8
contr.independent *(*contr.BayesTools*)*, 8
contr.meandif *(*contr.BayesTools*)*, 8
contr.orthonormal *(*contr.BayesTools*)*, 8

default_prior, 9
diagnostics, 10
diagnostics(), *37*
diagnostics_autocorrelation
        *(*diagnostics*)*, 10
diagnostics_density *(*diagnostics*)*, 10
diagnostics_trace *(*diagnostics*)*, 10
difftime, *34*

exp-aft, 11
exp_aft_density *(*exp-aft*)*, 11
exp_aft_hazard *(*exp-aft*)*, 11
exp_aft_log_density *(*exp-aft*)*, 11
exp_aft_log_hazard *(*exp-aft*)*, 11
exp_aft_log_survival *(*exp-aft*)*, 11
exp_aft_mean *(*exp-aft*)*, 11
exp_aft_p *(*exp-aft*)*, 11
exp_aft_q *(*exp-aft*)*, 11
exp_aft_r *(*exp-aft*)*, 11
exp_aft_sd *(*exp-aft*)*, 11
exp_aft_survival *(*exp-aft*)*, 11
Exponential, *26*, *30*
extract_flexsurv, 12

gamma-aft, 13
gamma_aft_density *(*gamma-aft*)*, 13
gamma_aft_hazard *(*gamma-aft*)*, 13
gamma_aft_log_density *(*gamma-aft*)*, 13
gamma_aft_log_hazard *(*gamma-aft*)*, 13
gamma_aft_log_survival *(*gamma-aft*)*, 13
gamma_aft_mean *(*gamma-aft*)*, 13
gamma_aft_p *(*gamma-aft*)*, 13
gamma_aft_q *(*gamma-aft*)*, 13
gamma_aft_r *(*gamma-aft*)*, 13
gamma_aft_sd *(*gamma-aft*)*, 13
gamma_aft_survival *(*gamma-aft*)*, 13
get_default_prior_aux *(*default_prior*)*, 9
get_default_prior_beta_alt
        *(*default_prior*)*, 9
get_default_prior_beta_alt,
        *(*default_prior*)*, 9
get_default_prior_beta_null
        *(*default_prior*)*, 9
get_default_prior_beta_null,
        *(*default_prior*)*, 9
get_default_prior_factor_alt
        *(*default_prior*)*, 9
get_default_prior_factor_alt,
        *(*default_prior*)*, 9
get_default_prior_factor_null
        *(*default_prior*)*, 9
get_default_prior_factor_null,
        *(*default_prior*)*, 9
get_default_prior_intercept
        *(*default_prior*)*, 9
get_default_prior_intercept,
        *(*default_prior*)*, 9

InvGamma, *26*, *30*

is.RoBSA, 14

llogis-aft, 14
llogis_aft_density *(*llogis-aft*)*, 14
llogis_aft_hazard *(*llogis-aft*)*, 14
llogis_aft_log_density *(*llogis-aft*)*, 14
llogis_aft_log_hazard *(*llogis-aft*)*, 14
llogis_aft_log_survival *(*llogis-aft*)*, 14
llogis_aft_mean *(*llogis-aft*)*, 14
llogis_aft_p *(*llogis-aft*)*, 14
llogis_aft_q *(*llogis-aft*)*, 14
llogis_aft_r *(*llogis-aft*)*, 14
llogis_aft_sd *(*llogis-aft*)*, 14
llogis_aft_survival *(*llogis-aft*)*, 14
lnorm-aft, 15
lnorm_aft_density *(*lnorm-aft*)*, 15
lnorm_aft_hazard *(*lnorm-aft*)*, 15
lnorm_aft_log_density *(*lnorm-aft*)*, 15
lnorm_aft_log_hazard *(*lnorm-aft*)*, 15
lnorm_aft_log_survival *(*lnorm-aft*)*, 15
lnorm_aft_mean *(*lnorm-aft*)*, 15
lnorm_aft_p *(*lnorm-aft*)*, 15
lnorm_aft_q *(*lnorm-aft*)*, 15
lnorm_aft_r *(*lnorm-aft*)*, 15
lnorm_aft_sd *(*lnorm-aft*)*, 15
lnorm_aft_survival *(*lnorm-aft*)*, 15
LocationScaleT, *26*, *30*
Lognormal, *26*, *30*

Normal, *26*, *30*

plot.prior(), *26*, *30*
plot.RoBSA, 16
plot_density *(*plot_prediction*)*, 20
plot_hazard *(*plot_prediction*)*, 20
plot_models, 18
plot_prediction, 20
plot_survival *(*plot_prediction*)*, 20
predict.RoBSA, 22
print.RoBSA, 24
print.summary.RoBSA, 24
prior, 25
prior(), *28*, *29*, *40*
prior_factor, 26
prior_informed, 28
prior_informed(), *30*
prior_informed_medicine_names, *29*, 30
prior_none, 30

RoBSA, 31, *35*
RoBSA(), *7*, *11*, *17*, *24*, *25*, *37*, *40*
RoBSA-package, 3
RoBSA.get_option *(*RoBSA_options*)*, 35
RoBSA.options *(*RoBSA_options*)*, 35
RoBSA.package *(*RoBSA-package*)*, 3
RoBSA_control, 34
RoBSA_options, 35
RoBSA_package *(*RoBSA-package*)*, 3

set_autofit_control *(*RoBSA_control*)*, 34
set_autofit_control(), *7*, *32*, *40*
set_autofit_control, *(*RoBSA_control*)*, 34
set_convergence_checks *(*RoBSA_control*)*, 34
set_convergence_checks(), *7*, *33*, *40*
summary.RoBSA, 36
summary.RoBSA(), *11*, *40*

update.RoBSA, *35*, 38

weibull-aft, 41
weibull_aft_density *(*weibull-aft*)*, 41
weibull_aft_hazard *(*weibull-aft*)*, 41
weibull_aft_log_density *(*weibull-aft*)*, 41
weibull_aft_log_hazard *(*weibull-aft*)*, 41
weibull_aft_log_survival *(*weibull-aft*)*, 41
weibull_aft_mean *(*weibull-aft*)*, 41
weibull_aft_p *(*weibull-aft*)*, 41
weibull_aft_q *(*weibull-aft*)*, 41
weibull_aft_r *(*weibull-aft*)*, 41
weibull_aft_sd *(*weibull-aft*)*, 41
weibull_aft_survival *(*weibull-aft*)*, 41