Package: bayesics (via r-universe)

October 14, 2026

Title Bayesian Analyses for One- and Two-Sample Inference and Regression Methods

Version 3.1.2

Maintainer Daniel K. Sewell <daniel-sewell@uiowa.edu>

Description Perform fundamental analyses using Bayesian parametric and non-parametric inference (regression, anova, 1 and 2 sample inference, non-parametric tests, etc.). (Practically) no Markov chain Monte Carlo (MCMC) is used; all exact finite sample inference is completed via closed form solutions or else through posterior sampling automated to ensure precision in interval estimate bounds. Diagnostic plots for model assessment, and key inferential quantities (point and interval estimates, probability of direction, region of practical equivalence, and Bayes factors) and model visualizations are provided. Bayes factors are computed either by the Savage Dickey ratio given in Dickey (1971) <doi:10.1214/aoms/1177693507> or by Chib's method as given in <doi:10.1080/01621459.1995.10476635>. Interpretations are from Kass and Raftery (1995) <doi:10.1080/01621459.1995.10476572>. ROPE bounds are based on discussions in Kruschke (2018) <doi:10.1177/2515245918771304>. Methods for determining the number of posterior samples required are described in Doss et al. (2014) <doi:10.1214/14-EJS957>. Bayesian model averaging is done in part by Feldkircher and Zeugner (2015) <doi:10.18637/jss.v068.i04>. Methods for contingency table analysis is described in Gunel et al. (1974) <doi:10.1093/biomet/61.3.545>. Variational Bayes (VB) methods are described in Salimans and Knowles (2013) <doi:10.1214/13-BA858>. Mediation analysis uses the framework described in Imai et al. (2010) <doi:10.1037/a0020761>. The loss-likelihood bootstrap used in the non-parametric regression modeling is described in Lyddon et al. (2019) <doi:10.1093/biomet/asz006>. Non-parametric survival methods are described in Qing et al. (2023) <doi:10.1002/pst.2256>. Methods used for the Bayesian Wilcoxon signed-rank analysis is

given in Chechile (2018) <doi:10.1080/03610926.2017.1388402> and for the Bayesian Wilcoxon rank sum analysis in Chechile (2020) <doi:10.1080/03610926.2018.1549247>. Correlation analysis methods are carried out by Barch and Chechile (2023) <doi:10.32614/CRAN.package.DFBA>, and described in Lindley and Phillips (1976) <doi:10.1080/00031305.1976.10479154> and Chechile and Barch (2021) <doi:10.1016/j.jmp.2021.102638>. See also Chechile (2020, ISBN: 9780262044585).

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Roxygen list(markdown = TRUE)

Suggests datasets, rstanarm, knitr, splines, testthat (>= 3.0.0)

Imports tidyr, dplyr, rlang, janitor, extraDistr, mvtnorm, Matrix, future, future.apply, ggplot2, patchwork, BMS, cluster, DFBA, tibble, survival, stringr

Config/testthat/edition 3

URL https://github.com/dksewell/bayesics

BugReports https://github.com/dksewell/bayesics/issues

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev

Repository https://dksewell.r-universe.dev

Date/Publication 2026-10-14 17:05:36 UTC

RemoteUrl https://github.com/dksewell/bayesics

RemoteRef HEAD

RemoteSha 679d65dcdbdd9b0d9004b7dead89c6090a9bbec7

Contents

aov_b . . . 3
b_procedure-class . . . 6
bayes_factors . . . 8
bayes_pvalue . . . 9
bma_inference . . . 12
case_control_b . . . 13
chisq_test_b . . . 15
coef . . . 17
cor_test_b . . . 18
credint . . . 20
find_beta_parms . . . 22
find_invgamma_parms . . . 23
frac_bayes_factors . . . 24

get_posterior_draws . . . 25
glm_b . . . 26
heteroscedasticity_test . . . 30
IC . . . 32
lm_b . . . 33
lm_b-class . . . 36
logLik.lm_b . . . 37
mediate_b . . . 38
negbinom . . . 41
np_glm_b . . . 41
plot . . . 44
plot_bands . . . 46
plot_dx . . . 48
poisson_test_b . . . 49
predict.lm_b . . . 51
print . . . 53
prop_test_b . . . 54
sign_test_b . . . 56
summary . . . 58
Surv . . . 59
survfit_b . . . 60
t_test_b . . . 62
vcov . . . 64
wilcoxon_test_b . . . 65

Index **68**

---

aov_b *Analysis of Variance Using Bayesian Methods*

Description

Analysis of Variance using Bayesian methods, including both proper and improper priors.

Usage

```
aov_b(
  formula,
  data,
  heteroscedastic = TRUE,
  prior_mean_mu,
  prior_mean_nu = 0.001,
  prior_var_shape = 0.001,
  prior_var_rate = 0.001,
  CI_level = 0.95,
  ROPE = 0.1,
  contrasts,
  improper = FALSE,
```

```
  seed = 1,
  mc_error = 0.002,
  compute_bayes_factor = TRUE
)
```

Arguments

formula	A formula specifying the model.
data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.
heteroscedastic	logical. Set to FALSE to assume all groups have equal variance.
prior_mean_mu	numeric. Hyperparameter for the a priori mean of the group means.
prior_mean_nu	numeric. Hyperparameter which scales the precision of the group means.
prior_var_shape	numeric. Twice the shape parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
prior_var_rate	numeric. Twice the rate parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
CI_level	numeric. Credible interval level.
ROPE	numeric. Used to compute posterior probability that Cohen's D +/- ROPE
contrasts	numeric/matrix. Either vector of length equal to the number of levels in the grouping variable, or else a matrix where each row is a separate contrast, and the number of columns match the number of levels in the grouping variable.
improper	logical. Should we use an improper prior that is proportional to the inverse of the variance?
seed	integer. Always set your seed!!!
mc_error	The number of posterior draws will ensure that with 99% probability the bounds of the credible intervals will be within $\pm$ mc_error$\times 4s_y$, that is, within 100mc_error% of the trimmed range of y. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.
compute_bayes_factor	logical. Computing the BF can be done analytically, but it requires an nxn matrix. If this will require more than 1GB of memory, compute_bayes_factor will automatically be set to FALSE. This setting can be overridden by setting compute_bayes_factor="force".

Details

MODEL: The likelihood model is given by

$$y_{gi} \overset{iid}{\sim} N(\mu_g, \sigma_g^2),$$

(although if heteroscedastic is set to FALSE, $\sigma_g^2 = \sigma_h^2 \; \forall g, h$).

The prior is given by

$$\mu_g|\sigma_g^2 \overset{iid}{\sim} N\left(\mu, \frac{\sigma_g^2}{\nu}\right), \sigma_g^2 \overset{iid}{\sim} \Gamma^{-1}(a/2, b/2),$$

where mu is set by `prior_mean_mu`, nu is set by `prior_mean_nu`, a is set by `prior_var_shape`, and b is set by `prior_var_rate`.

The posterior is

$$\mu_g|y, \sigma_g^2 \overset{iid}{\sim} N\left(\hat{\mu}_g, \frac{\sigma_g^2}{\nu_g}\right), \sigma_g^2|y \overset{iid}{\sim} \Gamma^{-1}(a_g/2, b_g/2),$$

where $\hat{\mu}_g$, ν_g, a_g, and b_g are all returned by `aov_b` in the named element `posterior_parameters`.

ROPE:

If missing, the ROPE bounds will be given under the principle of "half of a small effect size." Using Cohen's D of 0.2 as a small effect size, the ROPE is defined in terms of $-0.1 <$ Cohen's D < 0.1.

Value

Object of class `aov_b` and `lm_b`.

References

Charles R. Doss, James M. Flegal, Galin L. Jones, Ronald C. Neath "Markov chain Monte Carlo estimation of quantiles," Electronic Journal of Statistics, Electron. J. Statist. 8(2), 2448-2478, (2014)

Examples

```
# Create data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rep(letters[1:5],N/5))
test_data$outcome =
  rnorm(N,-1 + 2 * (test_data$x1 %in% c("d","e")) )

# Fit 1-way ANOVA model
fit1 <-
  aov_b(outcome ~ x1,
        test_data,
        prior_mean_mu = 2,
        prior_mean_nu = 0.5,
        prior_var_shape = 0.01,
        prior_var_rate = 0.01)
fit1
summary(fit1)
plot(fit1)
coef(fit1)
credint(fit1)
credint(fit1,
        CI_level = 0.99)
```

```
vcov(fit1)
fit1_predictions <-
  predict(fit1,
          CI_level = 0.99,
          PI_level = 0.9)
AIC(fit1)
BIC(fit1)
DIC(fit1)
WAIC(fit1)

# Implement contrasts
## One contrast
fit2 <-
  aov_b(outcome ~ x1,
        test_data,
        mc_error = 0.01,
        contrasts = c(-1/3,-1/3,-1/3,1/2,1/2))
fit2$contrasts
summary(fit2)
## Multiple contrasts
fit3 <-
  aov_b(outcome ~ x1,
        test_data,
        mc_error = 0.01,
        contrasts = rbind(c(-1/3,-1/3,-1/3,1/2,1/2),
                          c(-1/3,-1/3,-1/3,1,0)))
fit3$contrasts
summary(fit3)
```

`b_procedure-class` `b_procedure` *Objects*

Description

Objects of class `b_procedure` represent the result of a Bayesian procedure, including the data, prior specification, posterior summaries, and optional plotting output.

Details

A `b_procedure` object is a named list with the following components (not all bayesian procedures will yield objects with all of these entries):

name Character string giving the name of the procedure.

data A tibble containing the data used in the analysis.

print_data Logical; whether the data should be printed by `print.b_procedure`.

CI_level Numeric scalar giving the credible interval level as provided by the user.

sampling_design Character; Used by `independence_b`.

prior Character string describing the prior used.

results A tibble containing posterior summaries with columns:

- `Quantity`: character
- `Post Mean`: numeric
- `Lower`: numeric
- `Upper`: numeric
- `ROPE`: optional numeric
- `ROPE_lower_bound`,`ROPE_upper_bound`: optional numeric

PDir List containing:

- `description`: Character describing the probability of direction.
- `pdir`: Numeric scalar giving the probability of direction.

BF List containing:

- `description`: character
- `BF`: numeric scalar giving the Bayes factor
- `interpretation`: character

overall_ROPE List containing:

- `description`: character
- `Pr_in_ROPE`: numeric

plot A `ggplot` object associated with the procedure.

object_fit If applicable, the underlying fitted model object (e.g., `aov_b`).

notes character vector

S3 methods

The following methods are available for `lm_b` class objects: `print()` and `plot()`.

See Also

`print.b_procedure`, `plot.b_procedure`

Examples

```
cc_fit <- case_control_b(matrix(c(8,47,1,26),2,2))
cc_fit
plot(cc_fit)
```

bayes_factors *Bayes Factors*

Description

Bayes factors for for lm_b, glm_b, and survfit_b objects

Usage

```
bayes_factors(object, ...)

## S3 method for class 'lm_b'
bayes_factors(object, by = "coefficient", ...)

## S3 method for class 'glm_b'
bayes_factors(object, by = "coefficient", ...)

## S3 method for class 'survfit_b'
bayes_factors(object, object2, ...)
```

Arguments

object	lm_b, glm_b, or survfit_b object
...	Passed to methods.
by	character. Either "coefficient" or "variable". If the former, Bayes factors will be computed for each regression coefficient separately. If the latter, Bayes factors will be computed for each covariate separately.
object2	a second survfit_b object. Not used for other classes.

Details

Bayes factors are given in terms of favoring the two-tailed alternative hypothesis vs. the null hypothesis that the regression coefficient equals zero. Currently implemented for lm_b or glm_b objects. Note that for glm_b objects, if importance sampling was used, the model will be refit using fixed form variational Bayes to get the multivariate posterior density. The Bayes factor is then computed using the Savage-Dickey ratio.

Interpretation is taken from Kass and Raftery.

Value

A tibble with Bayes factors and interpretations.

References

Chib, S. (1995). Marginal Likelihood from the Gibbs Output. Journal of the American Statistical Association, 90(432), 1313–1321. https://doi.org/10.1080/01621459.1995.10476635

James M. Dickey. "The Weighted Likelihood Ratio, Linear Hypotheses on Normal Location Parameters." Ann. Math. Statist. 42 (1) 204 - 223, February, 1971. https://doi.org/10.1214/aoms/1177693507

Kass, R. E., & Raftery, A. E. (1995). Bayes Factors. Journal of the American Statistical Association, 90(430), 773–795.

Examples

```
# Generate some binomial data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome =
  rbinom(N,1,1.0 / (1.0 + exp(-(-2 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) ))))


# Fit a GLM
fit <-
  glm_b(outcome ~ x1 + x2 + x3,
        data = test_data,
        family = binomial(),
        seed = 2025)

# Compute the BF for each coefficient
bayes_factors(fit)

# Compute the BF for each variable
bayes_factors(fit,
              by = "variable")
```

`bayes_pvalue` *Bayesian P-values for Regression Models*

Description

Model diagnostics via Bayesian p-values for `lm_b` and `aov_b` objects

Usage

```
bayes_pvalue(object, statistic, mc_error, seed, ...)
```

```
## S3 method for class 'lm_b'
bayes_pvalue(object, statistic, mc_error = 0.005, seed = 1, ...)

## S3 method for class 'aov_b'
bayes_pvalue(object, statistic, mc_error = 0.005, seed = 1, ...)
```

Arguments

`object`	object of class `lm_b` or `aov_b`
`statistic`	Statistic used to compute Bayesian p-value. If missing, the default statistic will either be the Shapiro-Wilk test statistic if the family is `gaussian` or else the deviance. User specified functions are allowed, and must take in the response variable, its expected value, and if applicable to the family, dispersion (residual variance for `gaussian` and ϕ for `negbinom`, where $Var(y) = \mu + \mu^2/\phi$).
`mc_error`	The number of posterior draws will ensure that with 99% probability the estimated Bayesian p-value will be within $\pm$ `mc_error` of the actual Bayesian p-value. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.
`seed`	integer.
`...`	optional arguments.

Details

Overview:

Bayesian p-values are measures of how well the model match the data, as evaluated through the predictive posterior distribution. While they can be extremely flexible - testing very specific aspects of the model being fitted-, the default setting for `bayes_pvalue` is the deviance, acting to do an overall goodness-of-fit test. More generally, a Bayesian p-value takes a test statistic of the data and the model parameters $T(y, \theta)$ and compares the posterior probability that the test statistic evaluated at the observed data compared to data randomly generated according to the model. I.e., the Bayesian p-value is given by

$$\Pr(T(y_{obs}, \theta) > T(y_{pred}, \theta)|y_{pred}).$$

MC error:

The number of posterior samples is determined by the fact that if the true Bayesian p-value is p, the Monte Carlo estimate of the Bayesian p-value will have 99% probability of being with $\approx 2.3\sqrt{p(1-p)/L}$, where L is the number of posterior draws. The worst case scenario, in terms of MC variance is when the Bayesian p-value is $p = 0.5$. However, in such a case, there will be no question that the model fit is adequate and we can afford a much larger MC error. It is nearer thresholds (typically Bayesian p-values less than 0.05 or greater than 0.95 are cause for alarm) where we need more precise estimates of what the Bayesian p-value actually is, but at near these boundaries the MC error is much smaller than the worst case scenario of $p = 0.5$. Hence we compute the number of posterior samples to be within the user-specified `mc_error` at $p(1-p) = (0.15)(0.85)$.

The Monte Carlo error is implemented in such a way so as to obtain as few posterior samples as necessary to ensure a small probability of incorrectly determining an inadequate fit when the fit is good and vice versa.

Value

named list with:

- bpvalue - numeric between 0 and 1, giving the Bayesian p-value
- statistic_posterior_draws - tibble with two columns, one for $T(y_{obs}, \theta)$ and the other for $T(y_{pred}, \theta)$. See details.

Examples

```
# Create some data
set.seed(2026)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome =
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )

# Fit a linear regression model
fit =
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)

# Compute the Bayesian p-value based on the deviance
bp_deviance =
  bayes_pvalue(fit)
bp_deviance$bpvalue # We want a value near 0.5, say in (0.05,0.95).
plot(T_y_observed ~ T_y_predicted,
     data = bp_deviance$statistic_posterior_draws,
     xlab = expression(T(y[pred],theta)),
     ylab = expression(T(y[obs],theta)),
     pch = 16,
     cex = 0.1,
     col = gray(0.5,0.25))
abline(0,1,
       lwd = 2)

# Use a custom test statistic
bp_sw =
  bayes_pvalue(fit,
               statistic =
                 function(y,mu,dispersion){
                   shapiro.test((y - mu)/sqrt(dispersion))$statistic
                 }
  )
bp_sw$bpvalue
```

`bma_inference` *Bayesian Model Averaging*

Description

Estimates, CIs, and more from Bayesian Model Averaging.

Usage

```
bma_inference(
  formula,
  data,
  zellner_g = nrow(data),
  CI_level = 0.95,
  ROPE,
  mcmc_draws = 10000,
  n_models = 500,
  mc_error = 0.001,
  seed = 1,
  compute_residuals = TRUE,
  ...
)
```

Arguments

`formula`	A formula specifying the model.
`data`	Data used in linear regression model
`zellner_g`	numeric. Positive number giving the value of "g" in Zellner's g prior.
`CI_level`	Level for credible interval
`ROPE`	vector of positive values giving ROPE boundaries for each regression coefficient. Optionally, you can not include a ROPE boundary for the intercept. If missing, defaults go to those suggested by Kruchke (2018).
`mcmc_draws`	Integer. Number of draws passed into `bms`
`n_models`	Integer. The number of best models for which information is stored. See `bms` for more details.
`mc_error`	The number of posterior draws will ensure that with 99% probability the bounds of the credible intervals will be within $\pm$ `mc_error`$\times 4 s_y$, that is, within 100`mc_error`% of the trimmed range of y.
`seed`	Integer. Always set your seed!!!
`compute_residuals`	logical. Should residuals and standardized residuals be computed? It may be memory intensive for large datasets and small `mc_error`.
`...`	Other arguments for `bms`.

Details

bma_inference leverages the bms function from its eponymous R package, and then uses lm_b to obtain inference on the regression coefficients for Bayesian model averaging.

Value

Object of class lm_b_bma and lm_b.

References

Feldkircher, M. and S. Zeugner (2015): Bayesian Model Averaging Employing Fixed and Flexible Priors: The BMS Package for R, Journal of Statistical Software 68(4).

Examples

```
# Create data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5],
             x4 = rnorm(N),
             x5 = rnorm(N),
             x6 = rnorm(N),
             x7 = rnorm(N),
             x8 = rnorm(N),
             x9 = rnorm(N),
             x10 = rnorm(N))
test_data$outcome =
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )

# Fit linear model using Bayesian model averaging
fit <-
  bma_inference(outcome ~ .,
                test_data,
                user.int = FALSE)
summary(fit)
coef(fit)
credint(fit)
plot(fit)
```

---

case_control_b *Case-Control Analysis*

Description

Bayesian analysis of a case-control study (without covariates).

Usage

```
case_control_b(
  cases,
  controls,
  x,
  large_sample_approx,
  ROPE,
  prior_mean = 0,
  prior_sd = log(10)/1.96,
  plot = TRUE,
  CI_level = 0.95,
  seed = 1,
  mc_error = 0.005
)
```

Arguments

cases	vector of length 2, giving the numbers at risk and not at risk, respectively, for cases
controls	vector of length 2, giving the numbers at risk and not at risk, respectively, for controls
x	2x2 contingency table. The rows should depict the at risk status (first row is at risk, second row is not at risk), and the columns should depict the case control status (first column is case, second column is control).
large_sample_approx	If all cell counts of x are not too low (≥ 5) then use the approximation that the empirical log odds are normally distributed. (See details for more.) If missing, this will be set to TRUE iff all cell counts are greater than or equal to 5.
ROPE	ROPE for odds ratio. Provide either a single value or a vector of length two. If the former, the ROPE will be taken as (1/ROPE,ROPE). If the latter, these will be the bounds of the ROPE.
prior_mean	numeric. The prior mean on the log odds ratio. Defaults to 0 (i.e., odds ratio of 1).
prior_sd	numeric. The prior sd on the log odds ratio. Defaults to place 95% prior probability that the odds ratio is between 0.1 and 10.
plot	logical. Should a plot be shown?
CI_level	The posterior probability to be contained in the credible interval.
seed	integer. Always set your seed!!! (ignored if large_sample_approx = TRUE.)
mc_error	The relative monte carlo error of the quantiles of the CIs. (ignored if large_sample_approx = TRUE.) NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

If `large_sample_approx = TRUE` (the default if left missing and all cell counts are at least 5), then the likelihood is

$$\log(\hat{\omega}) \sim N\left(\log(\omega), \frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}\right),$$

where ω is the odds ratio, $\hat{\omega}$ is the empirical odds ratio, n_{ij}, $i, j = 1, 2$ are the cells of the 2x2 contingency table. The prior on $\log \omega$ is

$$\log \omega \sim N(\texttt{prior_mean}, \texttt{prior_sd}^2).$$

If the large sample approximation is not used, then inference is made on the odds ratio by instead putting uniform priors on $\Pr(exposure|outcome)$.

Value

An object of class `b_procedure-class`.

Examples

```
case_control_b(matrix(c(8,47,1,26),2,2))

case_control_b(c(8,47),
               c(1,26))
```

`chisq_test_b` *Test of Independence for 2-way Contingency Tables*

Description

Test independence assumption for 2-way contingency tables, where the data have been collected according to a multinomial, fixed row marginals, or fixed column marginals design.

Usage

```
independence_b(
  x,
  sampling_design = c("multinomial", "fixed rows", "fixed columns"),
  ROPE,
  prior = c("jeffreys", "uniform"),
  prior_shapes,
  CI_level = 0.95,
  seed = 1,
  mc_error = 0.002
)
```

Arguments

x	Either a table or a matrix of counts
sampling_design	Either "multinomial", "fixed rows", or "fixed columns"
ROPE	vector of positive values giving ROPE boundaries for each regression.
prior	Either "jeffreys" (Dirichlet(1/2)) or "uniform" (Dirichlet(1)). This is ignored if prior_shapes is provided.
prior_shapes	Either a single positive scalar, in which case a symmetric Dirichlet is used, or else a matrix matching the dimensions of x or a vector of length `prod(dim(x))`.
CI_level	The posterior probability to be contained in the credible interval.
seed	Always set your seed!
mc_error	This is the error in probability from the posterior CDF evaluated at the ROPE bounds. Note that if it is estimated that these probabilities are between 0.11 and 0.89, the more relaxed value of 0.01 is used. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

For a 2-way contingency table with R rows and C columns, evaluate the probability that

- the joint probabilities p_{ij} are all within the ROPE of $p_{i\cdot} \times p_{\cdot j}$ for `sampling_design = "multinomial"`
- the probabilities $p_{j|i}$ are all within the ROPE of $p_{\cdot j}$ if `sampling_design = "fixed rows"` or `"fixed columns"`

Value

An object of class `b_procedure-class`.

References

Gunel, Erdogan & Dickey, James (1974). Bayes factors for independence in contingency tables, Biometrika, 61(3), Pages 545–557, https://doi.org/10.1093/biomet/61.3.545

Kass, R. E., & Raftery, A. E. (1995). Bayes Factors. Journal of the American Statistical Association, 90(430), 773–795.

Examples

```
# Generate data
set.seed(2025)
N = 500
nR = 5
nC = 3
dep_probs =
  extraDistr::rdirichlet(1,rep(2,nR*nC)) |>
  matrix(nR,nC)
```

```
# Multinomial sampling
## Test independence
independence_b(round(N * dep_probs))

## Use other priors
independence_b(round(N * dep_probs),
               prior = "uniform")
independence_b(round(N * dep_probs),
               prior_shapes = 2)
independence_b(round(N * dep_probs),
               prior_shapes = matrix(1:(nR*nC),nR,nC))

# Fixed marginals
independence_b(round(N * dep_probs),
               sampling_design = "fixed rows")
independence_b(round(N * dep_probs),
               sampling_design = "fixed col")
```

---

coef *Coefficient Extraction for* bayesics *Objects*

Description

Extract the regression coefficients for lm_b and aov_b and return them as a vector.

Usage

```
## S3 method for class 'lm_b'
coef(object, ...)

## S3 method for class 'aov_b'
coef(object, ...)
```

Arguments

object	bayesics object
...	optional arguments.

Value

vector of coefficients

Examples

```
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rep(letters[1:5],N/5))
test_data$outcome =
  rnorm(N,-1 + 2 * (test_data$x1 %in% c("d","e")) )

# Fit 1-way ANOVA model
fit1 <-
  aov_b(outcome ~ x1,
        test_data,
        prior_mean_mu = 2,
        prior_mean_nu = 0.5,
        prior_var_shape = 0.01,
        prior_var_rate = 0.01)
coef(fit1)
```

`cor_test_b` *Test for Association/Correlation Between Paired Samples*

Description

Non-parametric test for association/correlation between paired samples via Kendall's tau, comparing the number of concordant and discordant pairs.

Usage

```
cor_test_b(x, ...)

## Default S3 method:
cor_test_b(
  x,
  y,
  tau = 0,
  ROPE,
  prior = c("centered", "uniform", "positive", "negative"),
  prior_shapes,
  CI_level = 0.95,
  plot = TRUE,
  ...
)

## S3 method for class 'formula'
cor_test_b(
  formula,
```

```
  data,
  tau = 0,
  ROPE,
  prior = "centered",
  prior_shapes,
  CI_level = 0.95,
  plot = TRUE,
  ...
)
```

Arguments

x, y	numeric vectors of data values. x and y must have the same length.
...	optional arguments.
tau	If provided, cor_test_b will return the posterior probability that Kendall's tau is less than this value.
ROPE	If a single number, ROPE will be $\tau\pm$ ROPE. If a vector of length 2, these will serve as the ROPE bounds. Defaults to ± 0.05.
prior	Beta prior used on phi (see details). Either "uniform' (Beta(1,1)), "centered' (Beta(2,2)), "positive" (Beta(3.9,2), putting 80% of the prior mass above 0.5), or "negative" (Beta(2,3.9), putting 80% of the prior mass below 0.5).
prior_shapes	Vector of length two, giving the shape parameters for the beta distribution that will act as the prior on ϕ (see details).
CI_level	The posterior probability to be contained in the credible interval.
plot	logical. Should a plot be shown?
formula	A formula of the form $u+v$, where each of u and v are numeric variables giving the data values for one sample.
data	A data.frame containing the variables in formula.

Details

cor_test_b relies on the robust Kendall's tau, defined to be

$$\tau := \frac{(\#\text{concordant pairs}) - (\#\text{discordant pairs})}{(\#\text{concordant pairs}) + (\#\text{discordant pairs})},$$

where a concordant pair is a pair of points such that if the rank of the x values is higher for the first (second) point of the pair, so too the rank of the y value is higher for the first (second) point of the pair.

The Bayesian approach of Chechile (2020) puts a Beta prior on phi, the proportion of concordance, i.e.,

$$\phi := \frac{(\#\text{concordant pairs})}{(\#\text{concordant pairs}) + (\#\text{discordant pairs})}.$$

The relationship between the two, then, is $\tau = 2\phi - 1$, or equivalently $\phi = (\tau + 1)/2$.

For more information, see dfba_bivariate_concordance and vignette("dfba_bivariate_concordance",package = "DFBA").

Value

An object of class b_procedure-class.

References

Chechile, R.A. (2020). Bayesian Statistics for Experimental Scientists: A General Introduction Using Distribution_Free Statistics. Cambridge: MIT Press.

Chechile, R.A., & Barch, D.H. (2021). A distribution-free, Bayesian goodness-of-fit method for assessing similar scientific prediction equations. Journal of Mathematical Psychology. https://doi.org/10.1016/j.jmp.2021.1026

Lindley, D. V., & Phillips, L. D. (1976). Inference for a Bernoulli process (a Bayesian view). The American Statistician, 30, 112-119.

Barch DH, Chechile RA (2023). DFBA: Distribution-Free Bayesian Analysis. doi:10.32614/CRAN.package.DFBA

Examples

```
# Generate data
set.seed(2025)
N = 50
x = rnorm(N)
y = x + 4 * rnorm(N)

# Test for non-zero correlation
cor_test_b(x,y)

# Input can be in the form of formula and data
cor_test_b(~ asdf + qwer,
           data = data.frame(asdf = x,
                             qwer = y))

# Other priors can be used, also.  See help for details.
cor_test_b(x,y,
           prior = "uniform")
cor_test_b(x,y,
           prior = "negative")
cor_test_b(x,y,
           prior = "positive")
cor_test_b(x,y,
           prior_shapes = c(10,10))
```

---

credint *Credible Intervals for Model Parameters*

Description

Computes credible intervals for one or more parameters in a fitted model.

Usage

```
credint(object, ...)

## S3 method for class 'lm_b'
credint(object, CI_level = 0.95, ...)

## S3 method for class 'aov_b'
credint(object, CI_level = 0.95, which = c("means", "pairwise"), ...)
```

Arguments

object	a fitted model object from bayesics
...	Passed to methods.
CI_level	the credible level required
which	character. For aov_b only. Either "means" (for the group means) or "pairwise" (for pairwise difference in means).

Value

Matrix of credible intervals

Examples

```
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rep(letters[1:5],N/5))
test_data$outcome =
  rnorm(N,-1 + 2 * (test_data$x1 %in% c("d","e")) )

# Fit 1-way ANOVA model
fit1 <-
  aov_b(outcome ~ x1,
        test_data,
        prior_mean_mu = 2,
        prior_mean_nu = 0.5,
        prior_var_shape = 0.01,
        prior_var_rate = 0.01)
credint(fit1)
```

`find_beta_parms` *Find Parameters for Beta Prior*

Description

Find parameters for Beta prior based on prior mean and one quantile.

Usage

```
find_beta_parms(
  mean,
  quantile,
  left_tail_prob,
  plot_results = TRUE,
  search_bounds = c(0.001, 100)
)
```

Arguments

`mean`	numeric between 0 and 1 giving the prior mean
`quantile`	numeric between 0 and 1 giving the quantile lying at left_tail_prob
`left_tail_prob`	numeric between 0 and 1 giving the prior probability of theta being less than or equal to quantile
`plot_results`	logical. Should the resulting inverse gamma distribution be plotted?
`search_bounds`	bounds with which to search. Sometimes you need to adjust this to get a good solution.

Value

Vector of beta shape parameters

Examples

```
find_beta_parms(2/5,0.68,0.9)
2/ (2 + 3)
qbeta(0.9,2,3)
```

`find_invgamma_parms` *Find Parameters for Inverse Gamma Prior*

Description

Find parameters for inverse gamma prior based on prior mean and one quantile, or, if in the context of linear regression, based on the a priori lower and upper bounds on the coefficient of determination

Usage

```
find_invgamma_parms(
  lower_quantile,
  upper_quantile,
  response_variance,
  lower_R2,
  upper_R2,
  probability,
  plot_results = TRUE
)
```

Arguments

`lower_quantile`	lower quantile desired
`upper_quantile`	upper quantile desired
`response_variance`	variance of the response variable of the regression model
`lower_R2, upper_R2`	We are a priori `probability` sure that the coefficient of determination (R^2) falls within these lower and upper bounds.
`probability`	prior probability to be contained within the lower and upper quantiles
`plot_results`	logical. Should the resulting inverse gamma distribution be plotted?

Details

Either provide the lower and upper quantiles that contain `probability` of the inverse gamma distribution, or if this is for linear regression, you can specify that you are a priori `probability` sure that the coefficient of determination (R^2) falls within the two bounds provided, assuming that the residual variance is $1 - R^2$ times the total variance.

Value

twice the shape and rate of the inverse gamma distribution.

Examples

```
# When aimed at linear regression via coefficient of determination...
hypothetical_s2_y = 2.0
lower_R2 = 0.05
upper_R2 = 0.85
find_invgamma_parms(response_variance = hypothetical_s2_y,
                    lower_R2 = lower_R2,
                    upper_R2 = upper_R2,
                    probability = 0.8)

# More arbitrary task...
find_invgamma_parms(0.3, # hypothetical_s2_y * (1.0 - upper_R2)
                    1.9, #hypothetical_s2_y * (1.0 - lower_R2)
                    probability = 0.8)
```

---

frac_bayes_factors *Fractional Bayes Factors*

Description

Compute fractional Bayes factors for `lm_b` objects

Usage

```
frac_bayes_factors(object1, object2, fractional_proportion)
```

Arguments

`object1`	object of class `lm_b`
`object2`	object of class `lm_b`
`fractional_proportion`	The fraction of the data used to create the prior in turn used to compute the marginal likelihood. By default, O'Hagan's recommendation of $max(log(n), ncol(X)+1)/n)$ is used.

Details

Fractional Bayes factors, devised by O'Hagan, are a way to use flat, even improper, priors to obtain valid Bayes factors. The idea is built on the notion of partial Bayes factors, where a part of the data is used to determine the prior, and the remaining is used to compare the models.

References

O'Hagan, Anthony. "Fractional Bayes Factors for Model Comparison." Journal of the Royal Statistical Society. Series B (Methodological), vol. 57, no. 1, 1995, pp. 99–138. https://doi.org/10.1111/j.2517-6161.1995.tb02017.x

Examples

```
set.seed(2026)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit_full <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
fit_no_x1 <-
  lm_b(outcome ~ x2 + x3,
       data = test_data)
fit_no_x2 <-
  lm_b(outcome ~ x1 + x3,
       data = test_data)

frac_bayes_factors(fit_full,
                   fit_no_x1)
frac_bayes_factors(fit_full,
                   fit_no_x2)
```

---

get_posterior_draws *Get Posterior Samples*

Description

Obtain a matrix of posterior samples for `lm_b` or `aov_b` objects

Usage

```
get_posterior_draws(object, n_draws, seed, ...)

## S3 method for class 'lm_b'
get_posterior_draws(object, n_draws = 10000, seed = 1, ...)

## S3 method for class 'aov_b'
get_posterior_draws(object, n_draws = 10000, seed = 1, ...)
```

Arguments

object	Object of class lm_b or aov_b
n_draws	integer. Number of posterior draws to obtain.
seed	integer.
...	optional arguments.

Value

matrix of posterior draws

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
pdraws <-
  get_posterior_draws(fit1)
```

---

glm_b *Bayesian Generalized Linear Models*

Description

glm_b is used to fit generalized linear models. It can be used to carry out regression for gaussian, binomial, poisson, and negative binomial data. Note that if the family is gaussian, this is just a wrapper for lm_b.

Usage

```
glm_b(
  formula,
  data,
  family,
  trials,
  prior = c("zellner", "normal", "improper"),
  zellner_g,
```

```
  prior_beta_mean,
  prior_beta_precision,
  prior_phi_mean = 1,
  ROPE,
  CI_level = 0.95,
  vb_maximum_iterations = 1000,
  algorithm = c("VB", "IS", "LSA"),
  proposal_df = 5,
  seed = 1,
  mc_error = 0.01,
  save_memory = FALSE
)
```

Arguments

formula	A formula specifying the model.
data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way. However, it is strongly recommended that you use this argument so that other generics for bayesics objects work correctly.
family	A description of the error distribution and link function to be used in the model. See ?glm for more information. Currently implemented families are binomial(), poisson(), negbinom(), and gaussian() (this last acts as a wrapper for lm_b. If missing family, glm_b will try to infer the data type; negative binomial will be used for count data.
trials	Either character naming the variable in data that corresponds to the number of trials in the binomial observations, or else an integer vector giving the number of trials for each observation.
prior	character. One of "zellner", "normal", or "improper", giving the type of prior used on the regression coefficients.
zellner_g	numeric. Positive number giving the value of "g" in Zellner's g prior. Ignored unless prior = "zellner". Default is the number of observations.
prior_beta_mean	numeric vector of same length as regression coefficients (denoted p). Unless otherwise specified, automatically set to rep(0,p). Ignored unless prior = "normal".
prior_beta_precision	pxp matrix giving a priori precision matrix to be scaled by the residual precision. Ignored unless prior = "normal".
prior_phi_mean	For negative binomial distributed outcomes, an exponential distribution is used for the prior of the dispersion parameter phi, parameterized such that Var$(y) = \mu + \frac{\mu^2}{\phi}$, so that the prior on ϕ is $\lambda e^{-\lambda\phi}$, where λ equals $1/$prior_phi_mean.
ROPE	vector of positive values giving ROPE boundaries for each regression coefficient. Optionally, you can not include a ROPE boundary for the intercept. If missing, defaults go to those suggested by Kruchke (2018).
CI_level	numeric. Credible interval level.

vb_maximum_iterations
: if algorithm = "VB", the number of iterations used in the fixed-form VB algorithm.

algorithm
: Either "VB" (default) for fixed-form variational Bayes, "IS" for importance sampling, or "LSA" for large sample approximation.

proposal_df
: degrees of freedom used in the multivariate t proposal distribution if algorithm = "IS".

seed
: integer. Always set your seed!!! Not used for algorithm = LSA.

mc_error
: If importance sampling is used, the number of posterior draws will ensure that with 99% probability the bounds of the credible intervals will be within $\pm$ mc_error. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

save_memory
: logical. If TRUE, a more memory efficient approach will be taken at the expense of computataional time (for important sampling only. But if memory is an issue, it's probably because you have a large sample size, in which case the normal approximation sans IS should probably work.)

Value

Object of class glm_b and lm_b.

Importance sampling:

glm_b will, unless use_importance_sampling = FALSE, perform importance sampling. The proposal will use a multivariate t distribution, centered at the posterior mode, with the negative hessian as its precision matrix. Do NOT treat the proposal_draws as posterior draws.

Priors:

If the prior is set to be either "zellner" or "normal", a normal distribution will be used as the prior of β, specifically

$$\beta \sim N(\mu, V)$$

where μ is the prior_beta_mean and V is the prior_beta_precision (not covariance) matrix.

- zellner: glm_b sets $\mu = 0$ and $V = \frac{1}{g} X^\top X$.
- normal: If missing prior_beta_mean, glm_b sets $\mu = 0$, and if missing prior_beta_precision V will be a diagonal matrix. The first element, corresponding to the intercept, will be $(2.5 \times \max \tilde{s}_y, 1)^{-2}$, where $\tilde{s}_y$ is max of 1 and the standard deviation of y. Remaining diagonal elements will equal $(2.5 s_y / s_x)^{-2}$, where s_x is the standard deviations of the covariates. This equates to being 95% certain a priori that a change in x by one standard deviation (s_x) would not lead to a change in the linear predictor of more than 5 standard deviations ($5 s_y$). This imposes weak regularization that adapts to the scale of the data elements.

ROPE:

If missing, the ROPE bounds will be given under the principle of "half of a small effect size."

- Gaussian. Using Cohen's D of 0.2 as a small effect size, the ROPE is built under the principle that moving the full range of X (i.e., $\pm 2 s_x$) will not move the mean of y by more than the overall mean of y minus $0.1 s_y$ to the overall mean of y plus $0.1 s_y$. The result is a ROPE equal to $|\beta_j| < 0.05 s_y / s_j$. If the covariate is binary, then this is simply $|\beta_j| < 0.2 s_y$.

- Poisson. FDA guidance suggests a small effect is a rate ratio less than 1.25. We use half this effect: 1.125, and consider ROPE to indicate that a moving the full range of X ($\pm 2s_x$ will not change the rate ratio by more than this amount. Thus the ROPE for the regression coefficient equals $|\beta| < \frac{\log(1.125)}{4s_x}$. For binary covariates, this is simply $|\beta| < \log(1.125)$.

References

Kruschke JK. Rejecting or Accepting Parameter Values in Bayesian Estimation. Advances in Methods and Practices in Psychological Science. 2018;1(2):270-280. doi:10.1177/2515245918771304

Tim Salimans. David A. Knowles. "Fixed-Form Variational Posterior Approximation through Stochastic Linear Regression." Bayesian Anal. 8 (4) 837 - 882, December 2013. https://doi.org/10.1214/13-BA858

Examples

```
# Generate some negative-binomial data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5],
             time = rexp(N))
test_data$outcome =
  rnbinom(N,
        mu = exp(-2 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e"))) * test_data$time,
          size = 0.7)

# Fit using variational Bayes (default)
fit_vb1 <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        seed = 2025)
fit_vb1
summary(fit_vb1,
        CI_level = 0.9)
plot(fit_vb1)
coef(fit_vb1)
credint(fit_vb1,
        CI_level = 0.99)
bayes_factors(fit_vb1,
              by = "v")
preds =
  predict(fit_vb1)

# Try different priors
fit_vb2 <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        seed = 2025,
```

```
        prior = "normal")
fit_vb2
fit_vb3 <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        seed = 2025,
        prior = "improper")
fit_vb3

# Use Importance sampling instead of VB
fit_is <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        algorithm = "IS",
        seed = 2025)
summary(fit_is)

# Use large sample approximation instead of VB
fit_lsa <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        algorithm = "LSA",
        seed = 2025)
summary(fit_lsa)
```

`heteroscedasticity_test` *Test for Heteroscedasticity in AOV Models*

Description

Use Chib's method to compute the Bayes factor to test for heteroscedasticity in analysis of variance models.

Usage

```
heteroscedasticity_test(hetero_model, homo_model)
```

Arguments

`hetero_model`	aov_b object where the heteroscedastic argument has been set to TRUE (the default)
`homo_model`	aov_b object where the heteroscedastic argument has been set to FALSE

Value

(returned invisible) A tibble with Bayes factors and interpretations.

References

Kass, R. E., & Raftery, A. E. (1995). Bayes Factors. Journal of the American Statistical Association, 90(430), 773–795.

Examples

```
# Test homoscedastic case
## Generate some data
set.seed(2025)
N = 200
test_data =
  data.frame(x1 = rep(letters[1:5],N/5))
test_data$outcome =
  rnorm(N,-1 + 2 * (test_data$x1 %in% c("d","e")) )

## Fit the anova models
hetero_model =
  aov_b(outcome ~ x1,
        test_data)
homo_model =
  aov_b(outcome ~ x1,
        test_data,
        heteroscedastic = FALSE)

## Perform test for heteroscedasticity using Bayes factors
heteroscedasticity_test(hetero_model,
                        homo_model)

# Test heteroscedastic case
## Generate some data
set.seed(2025)
N = 200
test_data =
  data.frame(x1 = rep(letters[1:5],N/5))
test_data$outcome =
  rnorm(N,
        -1 + 2 * (test_data$x1 %in% c("d","e")),
        sd = 3 - 2 * (test_data$x1 %in% c("d","e")))

## Fit the anova models
hetero_model =
  aov_b(outcome ~ x1,
        test_data)
homo_model =
  aov_b(outcome ~ x1,
        test_data,
        heteroscedastic = FALSE)
```

```
## Perform test for heteroscedasticity using Bayes factors
heteroscedasticity_test(hetero_model,
                        homo_model)
```

IC *Compute DIC or WAIC*

Description

Compute either Deviance Information Criterion (DIC) or Watanabe–Akaike Information Criterion (WAIC) for `aov_b` or `lm_b` objects. (Lower is better.)

Usage

```
DIC(object, ...)

## S3 method for class 'lm_b'
DIC(object, seed = 1, mc_error = 0.5, ...)

## S3 method for class 'aov_b'
DIC(object, ...)

## S3 method for class 'lm_b'
WAIC(object, seed = 1, mc_error = 0.5, ...)

## S3 method for class 'aov_b'
WAIC(object, ...)
```

Arguments

`object`	`aov_b`, `lm_b`, or `glm_b` object
`...`	Passed to methods.
`seed`	integer. Always set your seed!!!
`mc_error`	The number of posterior draws will ensure that with 99% probability the posterior mean of the deviance for DIC will be within $\pm$`mc_error`. For WAIC, this is based on extrapolating the standard error from the preliminary posterior samples and may be inaccurate (at least 2000 samples will be used in final calculation). NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

DIC and WAIC are computed via independent posterior sampling, ensuring that the final computed numbers is within `mc_error` of the actual DIC/WAIC with high probability.

Value

Numeric (or in the case of DIC, a numeric vector)

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
AIC(fit1)
BIC(fit1)
DIC(fit1)
WAIC(fit1)
```

---

lm_b *Bayesian Linear Models*

Description

lm_b is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although aov_b may provide a more convenient interface for ANOVA.)

Usage

```
lm_b(
  formula,
  data,
  weights,
  prior = c("zellner", "conjugate", "improper"),
  zellner_g,
  prior_beta_mean,
  prior_beta_precision,
  prior_var_shape,
  prior_var_rate,
  ROPE,
  CI_level = 0.95
)
```

Arguments

formula	A formula specifying the model.
data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way. However, it is strongly recommended that you use this argument so that other generics for bayesics objects work correctly.
weights	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, it is assumed that the variance of y_i can be written as $Var(y_i) = \sigma^2/w_i$. While the estimands remain the same, the estimation is done by performing unweighted lm_b on $W^{\frac{1}{2}}y$ and $W^{\frac{1}{2}}X$, where W is the diagonal matrix of weights. Note that this then affects the zellner prior.
prior	character. One of "zellner", "conjugate", or "improper", giving the type of prior used on the regression coefficients.
zellner_g	numeric. Positive number giving the value of "g" in Zellner's g prior. Ignored unless prior = "zellner".
prior_beta_mean	numeric vector of same length as regression coefficients (denoted p). Unless otherwise specified, automatically set to rep(0,p). Ignored unless prior = "conjugate".
prior_beta_precision	pxp matrix giving a priori precision matrix to be scaled by the residual precision.
prior_var_shape	numeric. Twice the shape parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
prior_var_rate	numeric. Twice the rate parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
ROPE	vector of positive values giving ROPE boundaries for each regression coefficient. Optionally, you can not include a ROPE boundary for the intercept. If missing, defaults go to those suggested by Kruchke (2018).
CI_level	numeric. Credible interval level.

Details

MODEL:

The likelihood is given by

$$y_i \overset{ind}{\sim} N(x_i'\beta, \sigma^2).$$

The prior is given by

$$\beta|\sigma^2 \sim N\left(\mu, \sigma^2 V^{-1}\right) \sigma^2 \sim \Gamma^{-1}(a/2, b/2).$$

- For Zellner's g prior, $\frac{1}{g}X'X$.
- The default for the conjugate prior is based on arguments from standardized regression. The default V is set according to the following: "a priori, we are 95% certain that a standard deviation increase in X will not lead to more than a 5 standard deviation in the mean of y." This leads to

$$V^{1/2} = diag(v^{1/2}, s_{x_1}, \ldots, s_{x_p})/2.5,$$

$$v^{-1} := \sum \frac{\bar{x}_j^2}{s_j^2}$$

where s_y is the standard deviation of y, and s_{x_j} is the standard deviation of the j^{th} covariate.

- Unless `prior_var_shape` AND `prior_var_rate` are provided, the inverse gamma prior on the residual variance will place 50% prior probability that the correlation between the response and the fitted values is between 0.1 and 0.9.
- If `prior = "improper"`, then the prior is

$$\pi(\beta, \sigma^2) \propto \frac{1}{\sigma^2}.$$

ROPE:

If missing, the ROPE bounds will be given under the principle of "half of a small effect size." Using Cohen's D of 0.2 as a small effect size, the ROPE is built under the principle that moving the full range of X (i.e., $\pm 2s_x$) will not move the mean of y by more than the overall mean of y minus $0.1s_y$ to the overall mean of y plus $0.1s_y$. The result is a ROPE equal to $|\beta_j| < 0.05s_y/s_j$. If the covariate is binary, then this is simply $|\beta_j| < 0.2s_y$.

Value

Object of class `lm_b`.

Examples

```
# Generate some data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome =
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )

# Find good hyperparameters for the residual variance
s2_hyperparameters =
  find_invgamma_parms(lower_R2 = 0.05,
                      upper_R2 = 0.95,
                      probability = 0.8,
                      response_variance = var(test_data$outcome))
## Check (should equal ~ 0.8)
extraDistr::pinvgamma((1.0 - 0.05) * var(test_data$outcome),
                      0.5 * s2_hyperparameters[1],
                      0.5 * s2_hyperparameters[2]) -
  extraDistr::pinvgamma((1.0 - 0.95) * var(test_data$outcome),
                        0.5 * s2_hyperparameters[1],
                        0.5 * s2_hyperparameters[2])

# Fit the linear model using a conjugate prior
fit1 <-
```

```
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data,
       prior = "conj",
       prior_var_shape = s2_hyperparameters["shape"],
       prior_var_rate = s2_hyperparameters["rate"])
fit1
summary(fit1,
        CI_level = 0.99)
plot(fit1)
coef(fit1)
credint(fit1,
        CI_level = 0.9)
bayes_factors(fit1)
bayes_factors(fit1,
              by = "var")
AIC(fit1)
BIC(fit1)
DIC(fit1)
WAIC(fit1)
vcov(fit1)
preds = predict(fit1)

# Try other priors
fit2 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data) # Default is prior = "zellner"
fit3 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data,
       prior = "improper")
```

`lm_b-class` `lm_b` *Objects*

Description

Objects of class `lm_b`

Details

An object of class `lm_b` contains at least the following:

summary Tibble giving the summary of the model parameters of having a minimum of:

`Variable` character
`Post Mean` Numeric

`Lower` Numeric

`Upper` Numeric

`Prob Dir` Numeric

`formula`

`data` A tibble containing the data used in the analysis.

`CI_level` Numeric scalar giving the credible interval level as provided by the user.

`fitted` Vector of fitted values

`residuals` Vector of Pearson residuals

`family`

`xlevels` Named list, giving the levels for each factor covariate.

`model_type` Character, either "parametric" or "nonparametric".

Further, one and only one of the following must be contained:

`posterior_covariance` matrix

`importance_sampling_weights`,`proposal_draws` Numeric and matrix, respectively

`posterior_draws` matrix

S3 methods

Methods are available for `print()` and `plot()`, depending on which components are present.

See Also

`print.b_procedure`, `plot.b_procedure`

Examples

```
## Not run:
cc_fit <- case_control_b(matrix(c(8,47,1,26),2,2))
cc_fit
plot(cc_fit)

## End(Not run)
```

---

logLik.lm_b *Extract Log-Likelihood*

Description

Computes the log-likelihood for fitted model objects.

Usage

```
## S3 method for class 'lm_b'
logLik(object, ...)
```

Arguments

object	An object of class aov_b, lm_b, glm_b, or lm_b_bma.
...	Further arguments passed to or from other methods.

Value

An object of class "logLik" with attributes "df" and "nobs".

See Also

logLik

---

mediate_b *Mediation using Bayesian Methods*

Description

Mediation analysis done using the potential outcomes framework outlined in Imai et al. (2010).

Usage

```
mediate_b(
  model_m,
  model_y,
  treat,
  control_value,
  treat_value,
  n_draws = 500,
  ask_before_full_sampling = TRUE,
  CI_level = 0.95,
  seed = 1,
  mc_error = ifelse("glm_b" %in% model_y, 0.01, 0.002),
  batch_size = 500
)
```

Arguments

model_m	a fitted model object of class lm_b or glm_b for mediator.
model_y	a fitted model object of class lm_b or glm_b for outcome.
treat	a character string indicating the name of the treatment variable used in the models. NOTE: Treatment variable must be numeric (even if it's 1's and 0's).

control_value	value of the treatment variable used as the control condition. Default is the 1st quintile of the treat variable.
treat_value	value of the treatment variable used as the treatment condition. Default is the 4th quintile of the treat variable.
n_draws	Number of preliminary posterior draws to assess final number of posterior draws required for accurate interval estimation
ask_before_full_sampling	logical. If FALSE, the user will not be asked if they want to complete the full sampling. Defaults to TRUE, as this can be a computationally intensive procedure.
CI_level	numeric. Credible interval level.
seed	integer. Always set your seed!!!
mc_error	positive scalar. The number of posterior samples will, with high probability, estimate the CI bounds up to $\pm$mc_error$\times$sd(y). NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.
batch_size	positive integer. Number of posterior draws to be taken at once. Higher values are more computationally intensive, but values which are too high might take up significant memory (allocates on the order of batch_size$\times$nrow(model_y$data)).

Details

The model is the same as that of Imai et al. (2010):

$$M_i(X) = w_i'\alpha_m + X\beta_m + \epsilon_{m,i}, y_i(X, M(\tilde{X})) = w_i'\alpha_y + X\beta_y + M(\tilde{X})\gamma + \epsilon_{y,i}, \epsilon_{m,i} \overset{iid}{\sim} N(0, \sigma_m^2), \epsilon_{y,i} \overset{iid}{\sim} N(0, \sigma_y^2),$$

where $M_i(X)$ is the mediator as a function of the treatment variable X, and w_i are confounder covariates.

Unlike the mediation R package, the estimation in mediate_b is fully Bayesian (as opposed to "quasi-Bayesian").

Value

A list with the following elements:

- summary - tibble giving results for causal mediation quantities
- posterior_draws (of counterfactual expectations)
- mc_error absolute error used, including any rescaling to match the scale of the outcome
- other inputs to mediate_b

References

Imai, Kosuke, et al. "A General Approach to Causal Mediation Analysis." Psychological Methods, vol. 15, no. 4, 2010, pp. 309–34, https://doi.org/10.1037/a0020761.

Examples

```
# Simplest case
## Generate some data
set.seed(2025)
N = 500
test_data =
  data.frame(tr = rnorm(N),
             x1 = rnorm(N))
test_data$m =
  rnorm(N, 0.4 * test_data$tr - 0.25 * test_data$x1)
test_data$outcome =
  rnorm(N,-1 + 0.6 * test_data$tr + 1.5 * test_data$m + 0.25 * test_data$x1)

## Fit the mediator and outcome models
m1 =
  lm_b(m ~ tr + x1,
       data = test_data)
m2 =
  lm_b(outcome ~ m + tr + x1,
       data = test_data)
## Estimate the causal mediation quantities
m3 <-
  mediate_b(m1,m2,
            treat = "tr",
            control_value = -2,
            treat_value = 2,
            n_draws = 500,
            mc_error = 0.05,
            ask_before_full_sampling = FALSE)
m3
summary(m3,
        CI_level = 0.9)

# More complicated scenario
## Generate some data
set.seed(2025)
N = 500
test_data =
  data.frame(tr = rep(0:1,N/2),
             x1 = rnorm(N))
test_data$m =
  rnorm(N, 0.4 * test_data$tr - 0.25 * test_data$x1)
test_data$outcome =
  rpois(N,exp(-1 + 0.6 * test_data$tr + 1.5 * test_data$m + 0.25 * test_data$x1))

## Fit the mediator and outcome models
m1 =
  lm_b(m ~ tr + x1,
       data = test_data)
m2 =
  glm_b(outcome ~ m + tr + x1,
        data = test_data,
```

```
        family = poisson())

##  Estimate the causal mediation quantities
m3 <-
  mediate_b(m1,m2,
            treat = "tr",
            control_value = 0,
            treat_value = 1,
            n_draws = 500,
            mc_error = 0.05,
            ask_before_full_sampling = FALSE)
summary(m3)
```

---

negbinom *Negative-Binomial Family*

Description

The negbinom() is an additional family to be considered alongside others documented under stats::family.

Usage

```
negbinom()
```

Value

an object of class "family". See stats::family.

Examples

```
negbinom()
```

---

np_glm_b *Non-parametric Linear Models*

Description

np_glm_b uses general Bayesian inference with loss-likelihood bootstrap. This is, as implemented here, a Bayesian non-parametric linear models inferential engine. Applicable data types are continuous (use family = gaussian()), count (use family = poisson()), or binomial (use family = binomial()).

Usage

```
np_glm_b(
  formula,
  data,
  family,
  loss = "selfinformation",
  loss_gradient,
  trials,
  n_draws,
  ask_before_full_sampling = TRUE,
  CI_level = 0.95,
  ROPE,
  seed = 1,
  mc_error = 0.01
)
```

Arguments

formula	A formula specifying the model.
data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way. However, it is strongly recommended that you use this argument so that other generics for bayesics objects work correctly.
family	A description of the error distribution and link function to be used in the model. See ?glm for more information. Currently implemented families are binomial(), poisson(), negbinom(), and gaussian() (this last acts as a wrapper for
loss	Either "selfinformation", or a function that takes in two arguments, the first of which should be the vector of outcomes and the second should be the expected value of y; The outcome of the function should be the loss evaluated for each observation. By default, the self-information loss is used (i.e., the negative log-likelihood). Note: I really do mean the expected value of y, even for binomial (i.e., n*p). If family = negbinom(), then a user-supplied loss function should take three arguments: y, mu, and phi, where phi is the dispersion parameter (i.e., $\text{Var}(y) = \mu + \mu^2/\phi$).
loss_gradient	If loss is a user-defined function (as opposed to "selfinformation"), supplying the gradient to the loss will speed up the algorithm.
trials	Integer vector giving the number of trials for each observation if family = binomial().
n_draws	integer. Number of posterior draws to obtain. If left missing, the large sample approximation will be used.
ask_before_full_sampling	logical. If TRUE, the user will be asked to specify whether they wish to commit to getting the full number of posterior draws to obtain precise credible interval bounds. Defaults to TRUE because the bootstrap is computationally intensive. Also, parallelization via future::plan is highly recommended for full sample.
CI_level	numeric. Credible interval level.

ROPE — vector of positive values giving ROPE boundaries for each regression coefficient. Optionally, you can not include a ROPE boundary for the intercept. If missing, defaults go to those suggested by Kruchke (2018).

seed — integer. Always set your seed!!!

mc_error — If large sample approximation is not used, the number of posterior draws will ensure that with 99% probability the bounds of the credible intervals will be within $\pm$ mc_error. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

Consider a population parameter of interest defined in terms of minimizing a loss function ℓ wrt the population distribution:

$$\theta(F_y) := \underset{\theta \in \Theta}{\text{argmax}} \int \ell(\theta, y) dF_y$$

If we use a non-parametric Dirichlet process prior on the distribution of y, F_y, and let the concentration parameter go to zero, we have the Bayesian bootstrap applied to a general Bayesian updating framework dictated by the loss function.

By default, the loss function is the self-information loss, i.e., the negative log likelihood. This then resembles a typical glm_b implementation, but is more robust to model misspecification.

Value

Object of class np_glm_b and lm_b.

References

S P Lyddon, C C Holmes, S G Walker, General Bayesian updating and the loss-likelihood bootstrap, Biometrika, Volume 106, Issue 2, June 2019, Pages 465–478, https://doi.org/10.1093/biomet/asz006

Examples

```
# Generate some data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome =
  rbinom(N,1,1.0 / (1.0 + exp(-(-2 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) ))))

# Fit the GLM via the (non-parametric) loss-likelihood bootstrap.
fit1 <-
  np_glm_b(outcome ~ x1 + x2 + x3,
           data = test_data,
           family = binomial())
fit1
```

```
summary(fit1,
        CI_level = 0.99)
plot(fit1)
coef(fit1)
credint(fit1)
predict(fit1,
        newdata = fit1$data[1,])
vcov(fit1)
```

---

plot *Plots* bayesics *Objects.*

Description

Create diagnostic plots, regression curve plots, and survival curves for lm_b, mediate_b, and survfit_b, and other procedure-specific plots for b_procedure objects.

Usage

```
## S3 method for class 'lm_b'
plot(
  x,
  type = c("diagnostics", "cred band", "pred band"),
  statistic,
  mc_error = 0.005,
  seed = 1,
  variable,
  exemplar_covariates,
  combine_pred_cred = TRUE,
  variable_seq_length = 30,
  return_as_list = FALSE,
  CI_level = 0.95,
  PI_level = 0.95,
  backtransformation = function(x) {
     x
 },
  ...
)

## S3 method for class 'mediate_b'
plot(
  x,
  type = c("diagnostics", "acme", "ade"),
  statistic = list(m = NULL, y = NULL),
  return_as_list = FALSE,
```

```
  seed = 1,
  mc_error = 0.005,
  ...
)

## S3 method for class 'survfit_b'
plot(x, n_draws = 10000, seed = 1, CI_level = 0.95, ...)

## S3 method for class 'b_procedure'
plot(x, ...)
```

Arguments

x	A bayesics object
type	character. Select any of "diagnostics", "cred band", and/or "pred band". If plotting a mediate_b object, the valid values for type are "diagnostics" (or "dx"), "acme", or "ade".
statistic	Statistic used to compute Bayesian p-value. If missing, the default statistic will either be the Shapiro-Wilk test statistic if the family is gaussian or else the deviance. User specified functions are allowed, and must take in the response variable, its expected value, and if applicable to the family, dispersion (residual variance for gaussian and ϕ for negbinom, where $Var(y) = \mu + \mu^2/\phi$). If x is of class mediate_b, statistic should be a named list with names equal to "m" and "y" for the mediator and the outcome models respectively.
mc_error	The number of posterior draws will ensure that with 99% probability the estimated Bayesian p-value will be within $\pm$ mc_error of the actual Bayesian p-value. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.
seed	integer.
variable	character. If type = "pdp" , which variable should be plotted?
exemplar_covariates	data.frame or tibble with exactly one row. Used to fix other covariates while varying the variable of interest for the plot.
combine_pred_cred	logical. If type includes both "cred band" and "pred band", should the credible band be superimposed on the prediction band or plotted separately?
variable_seq_length	integer. Number of points used to draw pdp.
return_as_list	logical. If TRUE, a list of ggplots will be returned, rather than a single plot produced by the patchwork package.
CI_level	Posterior probability covered by credible interval
PI_level	Posterior probability covered by prediction interval
backtransformation	function. If a transformation of the response variable was used, backtransformation should be the inverse of this transformation function. E.g., if you fit lm_b(log(y) ~ x), then set backtransformation=exp.

...	optional arguments.
n_draws	integer. Number of posterior draws used for visualization of survival curves. Ignored if x is not a survfit_b object.

Value

If return_as_list=TRUE, a list of requested ggplots.

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
plot(fit1)
```

---

plot_bands *Plot Credible and Prediction Bands*

Description

Plot credible and prediction bands for aov_b and lm_b, and credible bands for np_glm_b.

Usage

```
plot_bands(
  x,
  type,
  combine_pred_cred,
  CI_level,
  PI_level,
  backtransformation,
  return_as_list,
  ...
)

## S3 method for class 'lm_b'
plot_bands(
```

```
  x,
  type = c("cred band", "pred band"),
  combine_pred_cred = TRUE,
  CI_level = 0.95,
  PI_level = 0.95,
  backtransformation = function(x) {
     x
 },
  return_as_list = TRUE,
  variable,
  variable_seq_length = 30,
  exemplar_covariates,
  ...
)

## S3 method for class 'aov_b'
plot_bands(
  x,
  type = c("cred band", "pred band"),
  combine_pred_cred = TRUE,
  CI_level = 0.95,
  PI_level = 0.95,
  backtransformation = function(x) {
     x
 },
  return_as_list = TRUE,
  ...
)
```

Arguments

x	object of class aov_b and lm_b, or np_glm_b
type	character. Select "cred band", and/or "pred band". NOTE: the credible and prediction bands only work for numeric variables.
combine_pred_cred	logical. If type includes both "cred band" and "pred band", should the credible band be superimposed on the prediction band or plotted separately?
CI_level	Posterior probability covered by credible interval
PI_level	Posterior probability covered by prediction interval
backtransformation	function. If a transformation of the response variable was used, backtransformation should be the inverse of this transformation function. E.g., if you fit lm_b(log(y) ~ x), then set backtransformation=exp.
return_as_list	logical. If TRUE, a list of ggplots will be returned, rather than a single plot produced by the patchwork package.
...	arguments passed on to plot_bands
variable	character. If type = "pdp" , which variable should be plotted?

variable_seq_length
: integer. Number of points used to draw pdp.

exemplar_covariates
: data.frame or tibble with exactly one row. Used to fix other covariates while varying the variable of interest for the plot.

---

plot_dx *Diagnostic Plots for Bayesian Regression Objects*

Description

For regression objects based on a Gaussian distribution, qqplots and residuals-vs-fitted plots are provided. For all regression objects, a scatterplot of Bayesian p-values is provided (one hopes it will be cleanly bisected by the $y = x$ line).

Usage

```
plot_dx(x, statistic, mc_error, seed, return_as_list, ...)

## S3 method for class 'lm_b'
plot_dx(x, statistic, mc_error = 0.005, seed = 1, return_as_list = TRUE, ...)

## S3 method for class 'aov_b'
plot_dx(x, statistic, mc_error = 0.005, seed = 1, return_as_list = TRUE, ...)

## S3 method for class 'mediate_b'
plot_dx(
  x,
  statistic = list(m = NULL, y = NULL),
  mc_error = 0.005,
  seed = 1,
  return_as_list = TRUE,
  ...
)
```

Arguments

x
: object of class aov_b, lm_b, glm_b, or mediate_b

statistic
: Statistic used to compute Bayesian p-value. If missing, the default statistic will either be the Shapiro-Wilk test statistic if the family is gaussian or else the deviance. User specified functions are allowed, and must take in the response variable, its expected value, and if applicable to the family, dispersion (residual variance for gaussian and ϕ for negbinom, where $Var(y) = \mu + \mu^2/\phi$). If x is of class mediate_b, statistic should be a named list with names equal to "m" and "y" for the mediator and the outcome models respectively.

mc_error	The number of posterior draws will ensure that with 99% probability the estimated Bayesian p-value will be within ± mc_error of the actual Bayesian p-value. NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.
seed	integer.
return_as_list	logical. If TRUE, a list of ggplots will be returned, rather than a single plot produced by the patchwork package.
...	arguments passed on to plot_dx

---

poisson_test_b *Poisson Procedures*

Description

Make inference on one or two populations using Poisson distributed count data

Usage

```
poisson_test_b(
  x,
  offset,
  r,
  ROPE,
  prior = c("jeffreys", "flat"),
  prior_shape_rate,
  CI_level = 0.95,
  plot = TRUE,
  seed = 1,
  mc_error = 0.002
)
```

Arguments

x	Number of events. A vector of length one or two.
offset	Time, area, etc. measured in the Poisson process. NOTE: Do not take the log!
r	optional. If provided and inference is being made for a single population, poisson_test_b will return the posterior probability that the population rate is less than this value.
ROPE	ROPE for rate ratio if inference is being made for two populations. Provide either a single value or a vector of length two. If the former, the ROPE will be taken as (1/ROPE,ROPE). If the latter, these will be the bounds of the ROPE.
prior	Either "jeffreys" (Gamma(1/2,0)) or "flat" (Gamma(0.001,0.001)). This is ignored if prior_shape_rate is provided.

`prior_shape_rate`
: Vector of length two, giving the shape and rate parameters for the gamma distribution that will act as the prior on the population rates.

`CI_level`
: The posterior probability to be contained in the credible intervals.

`plot`
: logical. Should a plot be shown?

`seed`
: Always set your seed! (Unused for a single population rate)

`mc_error`
: The number of posterior draws will ensure that with 99% probability the bounds of the credible intervals of λ_1/λ_2 will be within $\pm$ `mc_error`. (Ignored for a single population rate.) NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

The likelihood is

$$y \sim Poi(\lambda t),$$

where λ is the rate, and t is the time or area observed and is given by the argument `offset`.

The prior is given by

$$\lambda \sim \Gamma(a, b),$$

where a and b are given by the argument `prior_shape_rate`. If `prior_shape_rate` is missing and `prior = "jeffreys"`, then a Jeffrey's prior will be used, i.e., $\Gamma(0.5, 0)$ (improper), while if `prior = "flat"`, $\Gamma(0.001, 0.001)$ will be used.

Value

An object of class `b_procedure-class`.

Examples

```
# One sample
poisson_test_b(x = 12)
## You can compute the posterior probability that the rate is less than r
poisson_test_b(x = 12,
               r = 8)

# Two samples
poisson_test_b(x = c(12,20))

# Offsets can be included:
poisson_test_b(x = c(12,20),
               offset = c(10,9))

# Different priors can be used
poisson_test_b(x = c(12,20),
               prior = "flat")
poisson_test_b(x = c(12,20),
               prior_shape_rate = c(20,1.5))
```

predict.lm_b *Predict Method for* bayesics *Model Fits*

Description

Get prediction point and interval estimates for objects of class aov_b, lm_b, np_glm_b, or lm_b_bma.

Usage

```
## S3 method for class 'lm_b'
predict(
  object,
  newdata,
  trials,
  CI_level = 0.95,
  PI_level = 0.95,
  seed = 1,
  n_draws = 5000,
  ...
)

## S3 method for class 'aov_b'
predict(object, CI_level = 0.95, PI_level = 0.95, ...)
```

Arguments

object	Object of class aov_b, lm_b, np_glm_b, or lm_b_bma
newdata	An optional data.frame in which to look for variables with which to predict.
trials	Integer vector giving the number of trials for each observation if family = binomial().
CI_level	Posterior probability covered by credible interval
PI_level	Posterior probability covered by prediction interval
seed	integer. Always set your seed!!!
n_draws	integer. Number of posterior draws used for prediction. Ignored if estimation method already relies on posterior sampling, in which case n_draws will match the number of posterior draws in the fitted object.
...	optional arguments.

Value

tibble with estimate (posterior mean), prediction intervals, and credible intervals for the mean.

Examples

```
# lm_b
## Create data
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )

## Fit linear model
fit <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
predict(fit)


# glm_b
## Generate some negative binomial data
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5],
             time = rexp(N))
test_data$outcome =
  rnbinom(N,
        mu = exp(-2 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e"))) * test_data$time,
          size = 0.7)

## Fit using variational Bayes (default)
fit_vb1 <-
  glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
        data = test_data,
        family = negbinom(),
        seed = 2025)
# Predict
predict(fit_vb1)

## Fit the GLM via the (non-parametric) loss-likelihood bootstrap.
fit_np <-
  np_glm_b(outcome ~ x1 + x2 + x3 + offset(log(time)),
           data = test_data,
           family = negbinom())
predict(fit_np)


# bma_inference
## Create data
```

```
set.seed(2025)
N = 500
test_data =
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5],
             x4 = rnorm(N),
             x5 = rnorm(N),
             x6 = rnorm(N),
             x7 = rnorm(N),
             x8 = rnorm(N),
             x9 = rnorm(N),
             x10 = rnorm(N))
test_data$outcome =
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )

## Fit linear model using Bayesian model averaging
fit <-
  bma_inference(outcome ~ .,
                test_data,
                user.int = FALSE)
predict(fit)
```

`print` *Print* `bayesics` *Objects.*

Description

Print `bayesics` objects, including objects of class `aov_b`, `lm_b`, `mediate_b`, `survfit_b`, and `b_procedure`.

Usage

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

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

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

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

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

Arguments

x	an object used to select a method.
...	optional arguments

Value

None

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
print(fit1)
```

---

prop_test_b *Binomial Procedures*

Description

prop_test_b either makes inference on a single population proportion, or else compares two population proportions.

Usage

```
prop_test_b(
  n_successes,
  n_failures,
  n_total,
  p,
  ROPE,
  prior = c("jeffreys", "uniform"),
  prior_shapes,
  CI_level = 0.95,
  plot = TRUE,
  seed = 1,
  mc_error = 0.002
)
```

Arguments

n_successes	integer/numeric vector of length 1 (for 1 population) or 2 (for 2 populations) providing the number of "successes"
n_failures	Similar to n_successes, but for failures. Only provide this OR n_total.
n_total	Similar to n_successes, but for total number of trials. Only provide this OR n_failures.
p	optional. If provided and inference is being made for a single population, prop_test_b will return the posterior probability that the population proportion is less than this value.
ROPE	ROPE for odds ratio if inference is being made for two populations. Provide either a single value or a vector of length two. If the former, the ROPE will be taken as (1/ROPE,ROPE). If the latter, these will be the bounds of the ROPE.
prior	Either "jeffreys" (Beta(1/2,1/2)) or "uniform" (Beta(1,1)). This is ignored if prior_shapes is provided.
prior_shapes	Vector of length two, giving the shape parameters for the beta distribution that will act as the prior on the population proportions.
CI_level	The posterior probability to be contained in the credible intervals.
plot	logical. Should a plot be shown?
seed	Always set your seed! (Unused for a single population proportion.)
mc_error	The number of posterior draws will ensure that with 99% probability the bounds of the credible intervals of $p_1 - p_2$ will be within $\pm$ mc_error. (Ignored for a single population proportion.) NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

The likelihood is given by

$$y \sim \text{Binom}(n, p),$$

and the prior on p is

$$p \sim Beta(a, b),$$

where a and b are given by the argument prior_shapes. If prior_shapes is missing and prior = "jeffreys", then a Jeffreys prior will be used ($Beta(1/2, 1/2)$), and if prior = "uniform", then a uniform prior will be used ($Beta(1, 1)$).

Value

An object of class b_procedure-class.

Examples

```
# Single population
prop_test_b(14,
            19)
# or another way of the same thing;
```

```
prop_test_b(14,
            n_total = 14 + 19)

# A null value compared against can be added:
prop_test_b(14,
            19,
            p = 0.5)

# Two populations
prop_test_b(c(14,22),
            c(19,45))
# or equivalently
prop_test_b(c(14,22),
            n_total = c(14,22) + c(19,45))
```

`sign_test_b` *Paired Sign Test*

Description

Non-parametric sign test for paired data.

Usage

```
sign_test_b(
  x,
  y,
  p0 = 0.5,
  prior = c("jeffreys", "uniform"),
  prior_shapes,
  ROPE,
  CI_level = 0.95,
  plot = TRUE
)
```

Arguments

`x`	Either numeric vector or binary vector. If the former, $z_i = 1_{[x_i > y_i]}$ if `y` is supplied, else $z_i = 1_{[x_i > 0]}$. If the latter, then $z_i = x_i$.
`y`	Optional numeric vector to pair with `x`.
`p0`	`sign_test_b` will return the posterior probability that $p < p0$. Defaults to 0.5, as is most typical in the sign test.
`prior`	Either "jeffreys" (Beta(1/2,1/2)) or "uniform" (Beta(1,1)). This is ignored if prior_shapes is provided.
`prior_shapes`	Vector of length two, giving the shape parameters for the beta distribution that will act as the prior on the probability that $z_i = 1$.

ROPE	positive numeric of length 1 or 2. If of length 1, then ROPE is taken to be $p0 \pm$ROPE. Defaults to ± 0.05.
CI_level	The posterior probability to be contained in the credible interval for p.
plot	logical. Should a plot be shown?

Details

The sign test looks at $z_i := 1_{[x_i > y_i]}$ rather than trying to model the distribution of (x_i, y_i). sign_test_b then uses the fact that

$$z_i \overset{iid}{\sim} Bernoulli(p)$$

and then makes inference on p. The prior on p is

$$p \sim Beta(a, b),$$

where a and b are given by the argument prior_shapes. If prior_shapes is missing and prior = "jeffreys", then a Jeffreys prior will be used ($Beta(1/2, 1/2)$), and if prior = "uniform", then a uniform prior will be used ($Beta(1, 1)$).

Value

An object of class b_procedure-class.

Examples

```
# Single population
sign_test_b(x = rnorm(50))

## Change ROPE
sign_test_b(x = rnorm(50),
            ROPE = 0.1)

## Change the prior
sign_test_b(x = rnorm(50),
            prior = "uniform")
sign_test_b(x = rnorm(50),
            prior_shapes = c(2,2))

## Two populations
sign_test_b(x = rnorm(50,1),
            y = rnorm(50,0))

## Change reference probability
sign_test_b(x = rnorm(50,1),
            y = rnorm(50,0),
            p0 = 0.7)
```

summary *Summarizing* bayesics *Fits*

Description

summary method for objects of class lm_b, aov_b, mediate_b, b_procedure, and survfit_b.

Usage

```
## S3 method for class 'lm_b'
summary(
  object,
  CI_level = 0.95,
  interpretable_scale = TRUE,
  print_results = TRUE,
  ...
)

## S3 method for class 'aov_b'
summary(object, CI_level = 0.95, print_results = TRUE, ...)

## S3 method for class 'mediate_b'
summary(object, CI_level = 0.95, print_results = TRUE, ...)

## S3 method for class 'b_procedure'
summary(object, ...)

## S3 method for class 'survfit_b'
summary(object, print_results = TRUE, ...)
```

Arguments

object	bayesics object
CI_level	Posterior probability covered by credible interval. Unused for b_procedure objects.
interpretable_scale	If a GLM is fit using binomial(link="logit"), poisson(link="log"), or negbinom(), and if interpretable_scale = TRUE then the results will be exponentiated.
print_results	logical
...	optional arguments for tibble::print.tbl_df.

Value

tibble with summary values

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
summary(fit1)
```

`Surv` *Create a Survival Object*

Description

Create a survival object, usually used as a response variable in a model formula. Argument matching is special for this function, see Details under `Surv`. This is a restricted wrapper around `Surv` and currently supports only right-censored data.

Usage

```
Surv(...)
```

Arguments

`...`	arguments to be passed into `survival::Surv`. Currently, the input must be of the form `Surv(time,event)` for right censored data.

Value

An object of class "`Surv`".

References

Therneau T (2024). A Package for Survival Analysis in R. R package version 3.8-3, https://CRAN.R-project.org/package=survival.

Examples

```
set.seed(2025)
N = 300
test_data =
  data.frame(outcome =
               rweibull(N,2,5))
test_data$observed =
  ifelse(test_data$outcome >= 7, 0, 1)
test_data$outcome =
  ifelse(dplyr::near(test_data$observed,1), test_data$outcome, 7)
Surv(test_data$outcome,
     test_data$observed)
```

---

survfit_b *Create Survival Curves*

Description

Use the semi-parametric piecewise exponential survival model to fit a survival curve to one or more samples

Usage

```
survfit_b(formula, data, prior_shape, prior_rate, max_n_time_bins, n_time_bins)
```

Arguments

formula	Either `Surv(time,event) ~ group` for multiple groups, or else `Surv(time,event) ~ 1` to make inference on a single population. The `event` variable must equal 1 if the event occurred and 0 if right censored. Currently right censoring is the only type of censoring allowed.
data	A data frame in which the variables specified in the formula will be found.
prior_shape	The shape parameter used in the gamma priors for the hazard rates
prior_rate	The rate parameter used in the gamma priors for the hazard rates
max_n_time_bins	integer. Maximum number of time bins, or "pieces", of the hazard function to be evaluated via Bayes factors. Ignored if `n_time_bins` is provided.
n_time_bins	Number of time bins used for hazard ratio. For a more data-driven approach, leave this argument missing and provide `max_n_time_bins`.

Details

The approach proposed by Qing et al. (2023) models the survival curve by way of piecewise exponential curves. That is, the hazard function is a piecewise function. The prior on the hazard within each "piece", or equivalently the rate of the exponential distribution, is a conjugate gamma distribution. Unless specified, the prior shape and rate for each piece is the posterior under the assumption that the data follow a single exponential distribution.

Unless prespecified by the user, the number of breaks in the hazard function is determined by Bayes factors, which can be quickly computed analytically.

If more than one population is being compared, then as before Bayes factors will be used to determine the number of breaks in each group's hazard function, and then Bayes factors will be used to compare the hypothesis that each group has a separate survival function vs. the null hypothesis that all groups share the same survival function.

Value

Object of class `survfit_b` with the following:

- `posterior_parameters` An `n_time_bins`x2 matrix whose columns provide shapes and rates of the gamma posterior distribution of each of the piecewise hazard rates.
- `intervals` An `n_time_bins`x2 matrix whose columns provide the start and endpoints of each time bin. If comparing multiple samples, a list of such matrices will be provided.
- `marginal_likelihood`
- `data`

If comparing multiple samples, each group will have a list of `posterior_parameters` and `intervals`.

References

Qing Y, Thall PF, Yuan Y. A Bayesian piecewise exponential phase II design for monitoring a time-to-event endpoint. Pharm Stat. 2023 Jan;22(1):34-44. doi: 10.1002/pst.2256. Epub 2022

Examples

```
# Single population
set.seed(2025)
N = 300
test_data =
  data.frame(outcome =
               rweibull(N,2,5))
test_data$observed =
  ifelse(test_data$outcome >= 7, 0, 1)
test_data$outcome =
  ifelse(dplyr::near(test_data$observed,1), test_data$outcome, 7)
fit1 =
  survfit_b(Surv(test_data$outcome,
                 test_data$observed) ~ 1)
fit1
plot(fit1)
```

```
# Multiple populations
set.seed(2025)
N = 300
test_data =
  data.frame(outcome =
               c(rweibull(2*N/3,2,5),
                 rweibull(N/3,2,10)),
             x1 = rep(letters[1:3],each = N/3))
test_data$observed =
  ifelse(test_data$outcome >= 9, 0, 1)
test_data$outcome =
  ifelse(dplyr::near(test_data$observed,1), test_data$outcome, 9)
fit2 =
  survfit_b(Surv(outcome,
                 observed) ~ x1,
            data = test_data)
fit2
plot(fit2)
```

---

t_test_b *t-test*

Description

One and two sample t-tests on vectors of data

Usage

```
t_test_b(
  x,
  y,
  mu,
  paired = FALSE,
  data,
  heteroscedastic = TRUE,
  prior_mean_mu,
  prior_mean_nu = 0.001,
  prior_var_shape = 0.001,
  prior_var_rate = 0.001,
  CI_level = 0.95,
  ROPE = 0.1,
  improper = FALSE,
  plot = TRUE,
  seed = 1,
  mc_error = 0.002
)
```

Arguments

x	Either a (non-empty) numeric vector of data values, or a formula of the form outcome ~ grouping variable.
y	an optional (non-empty) numeric vector of data values
mu	optional. If supplied, t_test_b will return the posterior probabilty that the population mean (ignored in 2 sample inference) is less than this value.
paired	logical. If TRUE, provide both x and y as vectors.
data	logical. Only used if x is a formula.
heteroscedastic	logical. Set to FALSE to assume all groups have equal variance.
prior_mean_mu	numeric. Hyperparameter for the a priori mean of the group means.
prior_mean_nu	numeric. Hyperparameter which scales the precision of the group means.
prior_var_shape	numeric. Twice the shape parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
prior_var_rate	numeric. Twice the rate parameter for the inverse gamma prior on the residual variance(s). I.e., $\sigma^2 \sim IG$(prior_var_shape/2,prior_var_rate/2).
CI_level	numeric. Credible interval level.
ROPE	numeric. Used to compute posterior probability that Cohen's D +/- ROPE
improper	logical. Should we use an improper prior that is proportional to the inverse of the variance?
plot	logical. Should the resulting inverse gamma distribution be plotted?
seed	integer. Always set your seed!!!
mc_error	The number of posterior draws will ensure that with 99% probability the bounds of the credible intervals will be within $\pm$ mc_error$\times 4s_y$, that is, within 100mc_error% of the trimmed range of y. (Ignored for single population inference.) NOTE: Cross-platform reproducibility may not be guaranteed due to numerical differences in stats::density() between operating systems, leading to slightly different numbers of posterior samples obtained.

Details

A one and two sample t-test is nothing more than a special case of one-way anova. See aov_b for details.

Value

An object of class b_procedure-class.

Examples

```
# Single population
t_test_b(rnorm(50))
# or an alternative input format
```

```
t_test_b(outcome ~ 1,
         data = data.frame(outcome = rnorm(50)))

# Two populations
t_test_b(rnorm(50),
         rnorm(15,1))

# or an alternative input format
t_test_b(outcome ~ group_variable,
         data =
           data.frame(outcome = c(rnorm(50),
                                  rnorm(15,1)),
                      group_variable = rep(c("a","b"),
                                           c(50,15))))
```

---

vcov *Calculate Posterior Variance-Covariance Matrix for a Bayesian Fitted Model Object*

Description

Returns the variance-covariance matrix of the parameters of a fitted model object. Note that if the family is `negbinom`, then the final row/column of this matrix will correspond to the log dispersion parameter.

Usage

```
## S3 method for class 'lm_b'
vcov(object, ...)
```

Arguments

`object`	a fitted model object from `bayesics`.
`...`	Passed to methods.

Value

A matrix of the covariance matrix for the regression coefficients. If the posterior is a multivariate t distribution (or consists of independent t's in the case of heteroscedastic 1-way ANOVA), the degrees of freedom are returned as the `df` attribute of the matrix. Note that for `lm_b` and `aov_b` objects, this function already takes into account the uncertainty around the residual variance.

Examples

```
set.seed(2025)
N = 500
test_data <-
  data.frame(x1 = rnorm(N),
             x2 = rnorm(N),
             x3 = letters[1:5])
test_data$outcome <-
  rnorm(N,-1 + test_data$x1 + 2 * (test_data$x3 %in% c("d","e")) )
fit1 <-
  lm_b(outcome ~ x1 + x2 + x3,
       data = test_data)
vcov(fit1)
```

`wilcoxon_test_b` *Bayesian Wilcoxon Rank Sum (aka Mann-Whitney U) and Signed Rank Analyses*

Description

Non-parametric tests comparing one or two samples via an implementation of the Bayesian Wilcoxon rank sum (aka Mann-Whitney U) and signed rank analyses. The analyses are being performed by dfba_wilcoxon and dfba_mann_whitney, with additional posterior quantities of interest provided here.

Usage

```
wilcoxon_test_b(
  x,
  y,
  paired = FALSE,
  p = 0.5,
  ROPE,
  prior = c("centered", "uniform"),
  prior_shapes,
  CI_level = 0.95,
  plot = TRUE,
  seed = 1
)
```

Arguments

x	numeric vector of data values. Non-finite (e.g., infinite or missing) values will be omitted.
y	an optional numeric vector of data values: as with x non-finite values will be omitted.

paired	if TRUE and y is supplied, x-y will be the input of the Bayesian Wilcoxon signed rank test.
p	numeric. • Signed rank: wilcox_test_b will return the posterior probability that the population proportion of positive values (i.e., $x > y$) is greater than this value. • Rank sum/Mann-Whitney U: wilcox_test_b will return the posterior probability that the Ω_x (see details) is greater than this value.
ROPE	If a single number, ROPE will be p$\pm$ROPE. If a vector of length 2, these will serve as the ROPE bounds. Defaults to ± 0.05.
prior	Prior used on the probability that x > y. Either "uniform" (Beta(1,1)), or "centered" (Beta(2,2)). This is ignored if prior_shapes is provided.
prior_shapes	Vector of length two, giving the shape parameters for the beta distribution that will act as the prior on the population proportions.
CI_level	The posterior probability to be contained in the credible interval.
plot	logical. Should a plot be shown?
seed	Always set your seed! (Unused for ≥ 20 observations.)

Details

Bayesian Wilcoxon signed rank analysis For a single input vector or paired data, the Bayesian signed rank analysis will be performed. The estimand is the proportion of (differenced) values that are positive. For more information, see dfba_wilcoxon and vignette("dfba_wilcoxon",package = "DFBA").

Bayesian Wilcoxon rank sum/Mann-Whitney analysis For unpaired x and y inputs, the Bayesian rank sum analysis will be performed. The estimand is $\Omega_x := \lim_{n\to\infty} \frac{U_x}{U_x+U_y}$, where U_x is the number of pairs (i, j) such that $x_i > y_j$, and vice versa for U_y. That is, it is the population proportion of all untied pairs for which $x > y$. Larger values imply that x is stochastically larger than y. For more information, see dfba_mann_whitney and vignette("dfba_mann_whitney",package = "DFBA").

Value

An object of class b_procedure-class.

References

Chechile, R.A. (2020). Bayesian Statistics for Experimental Scientists: A General Introduction to Distribution-Free Methods. Cambridge: MIT Press.

Chechile, R. A. (2018) A Bayesian analysis for the Wilcoxon signed-rank statistic. Communications in Statistics - Theory and Methods, https://doi.org/10.1080/03610926.2017.1388402

Chechile, R.A. (2020). A Bayesian analysis for the Mann-Whitney statistic. Communications in Statistics – Theory and Methods 49(3): 670-696. https://doi.org/10.1080/03610926.2018.1549247.

Barch DH, Chechile RA (2023). DFBA: Distribution-Free Bayesian Analysis. doi:10.32614/CRAN.package.DFBA

Examples

```
# Signed rank analysis
## Generate data
N = 150
set.seed(2025)
test_data =
  data.frame(x = rbeta(N,2,10),
             y = rbeta(N,5,10))

## input differenced data
wilcoxon_test_b(test_data$x - test_data$y)
## input paired data vectors individually
wilcoxon_test_b(test_data$x,
                test_data$y,
                paired = TRUE)

## Use different priors
wilcoxon_test_b(test_data$x - test_data$y,
                prior = "uniform")
wilcoxon_test_b(test_data$x - test_data$y,
                prior_shapes = c(5,5))

## Change ROPE bounds
wilcoxon_test_b(test_data$x - test_data$y,
                ROPE = 0.1)

# Rank sum analysis
## Generate data
set.seed(2025)
N = 150
x = rbeta(N,2,10)
y = rbeta(N + 1,5,10)

## Perform analysis
wilcoxon_test_b(x,y)
```

Index

aov_b, 3, *63*

b_procedure-class, 6
bayes_factors, 8
bayes_pvalue, 9
bma_inference, 12
bms, *12*

case_control_b, 13
chisq_test_b, 15
coef, 17
cor_test_b, 18
credint, 20

dfba_bivariate_concordance, *19*
dfba_mann_whitney, *65, 66*
dfba_wilcoxon, *65, 66*
DIC *(*IC*)*, 32

find_beta_parms, 22
find_invgamma_parms, 23
frac_bayes_factors, 24

get_posterior_draws, 25
glm, *27*, *42*
glm_b, 26

heteroscedasticity_test, 30

IC, 32
independence_b *(*chisq_test_b*)*, 15

lm_b, *5*, *13*, *28*, 33, *35*, *43*
lm_b-class, 36
logLik, *38*
logLik.lm_b, 37

mediate_b, 38

negbinom, 41
np_glm_b, 41

plot, 44
plot.b_procedure, *7*, *37*
plot_bands, 46
plot_dx, 48
poisson_test_b, 49
predict.aov_b *(*predict.lm_b*)*, 51
predict.lm_b, 51
print, 53
print.b_procedure, *7*, *37*
prop_test_b, 54

sign_test_b, 56
summary, 58
Surv, 59, *59*
survfit_b, 60

t_test_b, 62

var_test_b *(*heteroscedasticity_test*)*, 30
vcov, 64

WAIC *(*IC*)*, 32
wilcoxon_test_b, 65