Package 'priorsense'

November 1, 2024

Title Prior Diagnostics and Sensitivity Analysis

Version 1.0.4

Description Provides functions for prior and likelihood sensitivity analysis in Bayesian models. Currently it implements methods to determine the sensitivity of the posterior to power-scaling perturbations of the prior and likelihood.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports checkmate (>= 2.3.1), ggdist (>= 3.3.2), ggh4x (>= 0.2.5), ggplot2 (>= 3.5.1), matrixStats (>= 1.3.0), methods, posterior (>= 1.6.0), rlang (>= 1.1.4), stats, tibble (>= 3.2.1), utils

Suggests bayesplot (>= 1.11.1), brms (>= 2.22.0), cmdstanr (>= 0.8.1), iwmm (>= 0.0.1), knitr (>= 1.47), philentropy (>= 0.8.0), rstan (>= 2.32.6), testthat (>= 3.0.0), transport (>= 0.15), rmarkdown (>= 2.27)

Config/testthat/edition 3

Depends R (>= 3.6.0)

VignetteBuilder knitr

Additional_repositories https://topipa.r-universe.dev, https://stan-dev.r-universe.dev

URL https://n-kall.github.io/priorsense/

NeedsCompilation no

Author Noa Kallioinen [aut, cre, cph], Topi Paananen [aut], Paul-Christian Bürkner [aut], Aki Vehtari [aut], Frank Weber [ctb]

Maintainer Noa Kallioinen <noa.kallioinen@aalto.fi>

Repository CRAN

Date/Publication 2024-11-01 12:30:02 UTC

Contents

`priorsense-package` *priorsense: Prior (and likelihood) diagnostics and sensitivity analysis*

Description

The **priorsense** package provides functions for prior and likelihood sensitivity analysis of Bayesian models. Currently it implements methods to determine the sensitivity of the posterior to power-scaling perturbations of the prior and likelihood.

Details

The main diagnostic function provided by **priorsense** is `powerscale_sensitivity`. Given a fitted model or draws object, it computes the powerscaling sensitivity diagnostic described in Kallioinen et al. (2023). It does so by perturbing the prior and likelihood and computing the effect on the posterior, without needing to refit the model (using Pareto smoothed importance sampling and importance weighted moment matching; Vehtari et al. 2022, Paananen et al. 2021).

In addition, visual diagnostics are available by first using `powerscale_sequence` to create a sequence of perturbed posteriors, and then a plot function such as `powerscale_plot_ecdf` to visualise the change.

The following global options are available:

- `priorsense.plot_help_text`: If `TRUE` (the default), priorsense plots will include a title and explanatory text. If `FALSE` they will not.

Author(s)

Maintainer: Noa Kallioinen <noa.kallioinen@aalto.fi> [copyright holder]

Authors:

- Topi Paananen
- Paul-Christian Bürkner

- Aki Vehtari

Other contributors:

- Frank Weber [contributor]

References

Kallioinen, N., Paananen, T., Bürkner, P-C., Vehtari, A. (2023). Detecting and diagnosing prior and likelihood sensitivity with power-scaling perturbations. *Statistics and Computing*. 34(57). doi:10.1007/s11222-023-10366-5

Vehtari, A., Simpson, D., Gelman, A., Yao, Y., and Gabry, J. (2024). Pareto smoothed importance sampling. *Journal of Machine Learning Research*. 25(72). https://jmlr.org/papers/v25/19-556.html

Paananen, T., Piironen, J., Bürkner, P-C., Vehtari, A. (2021). Implicitly adaptive importance sampling. *Statistics and Computing*. 31(16). doi:10.1007/s11222-020-09982-2

See Also

powerscale_sensitivity powerscale_sequence powerscale powerscale_plot_ecdf powerscale_plot_dens powerscale_plot_quantities

---

cjs_dist *Cumulative Jensen-Shannon divergence*

Description

Computes the cumulative Jensen-Shannon distance between two samples.

Usage

```
cjs_dist(
  x,
  y,
  x_weights = NULL,
  y_weights = NULL,
  metric = TRUE,
  unsigned = TRUE,
  ...
)
```

Arguments

x	numeric vector of samples from first distribution
y	numeric vector of samples from second distribution
x_weights	numeric vector of weights of first distribution
y_weights	numeric vector of weights of second distribution

`metric`	Logical; if TRUE, return square-root of CJS
`unsigned`	Logical; if TRUE then return max of CJS(P(x) ‖ Q(x)) and CJS(P(-x) ‖ Q(-x)). This ensures invariance to transformations such as PCA.
`...`	unused

Details

The Cumulative Jensen-Shannon distance is a symmetric metric based on the cumulative Jensen-Shannon divergence. The divergence CJS(P ‖ Q) between two cumulative distribution functions P and Q is defined as:

$$CJS(P||Q) = \sum P(x) \log \frac{P(x)}{0.5(P(x) + Q(x))} + \frac{1}{2 \ln 2} \sum (Q(x) - P(x))$$

The symmetric metric is defined as:

$$CJS_{dist}(P||Q) = \sqrt{CJS(P||Q) + CJS(Q||P)}$$

This has an upper bound of $\sqrt{\sum (P(x) + Q(x))}$

Value

distance value based on CJS computation.

References

Nguyen H-V., Vreeken J. (2015). Non-parametric Jensen-Shannon Divergence. In: Appice A., Rodrigues P., Santos Costa V., Gama J., Jorge A., Soares C. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2015. Lecture Notes in Computer Science, vol 9285. Springer, Cham. doi:10.1007/978-3-319-23525-7_11

Examples

```
x <- rnorm(100)
y <- rnorm(100, 2, 2)
cjs_dist(x, y, x_weights = NULL, y_weights = NULL)
```

`create-priorsense-data`

Create data structure for priorsense

Description

Create a data structure that contains all required data and functions for priorsense

Usage

```
create_priorsense_data(x, ...)

## Default S3 method:
create_priorsense_data(
  x,
  fit = NULL,
  log_prior_fn = log_prior_draws,
  log_lik_fn = log_lik_draws,
  log_prior = NULL,
  log_lik = NULL,
  log_ratio_fn = NULL,
  ...
)

## S3 method for class 'stanfit'
create_priorsense_data(x, ...)

## S3 method for class 'CmdStanFit'
create_priorsense_data(x, ...)

## S3 method for class 'draws'
create_priorsense_data(x, ...)
```

Arguments

x	an object for which the method is defined
...	arguments passed to methods
fit	a model fit object (only used if x is not a fit object)
log_prior_fn	function to derive log prior from object
log_lik_fn	function to derive log likelihood from object
log_prior	draws from log prior
log_lik	draws from log likelihood
log_ratio_fn	function for moment matching

Value

A priorsense_data object, which contains the data and functions to run sensitivity analyses.

Examples

```
x <- example_powerscale_model()
drw <- x$draws

psd <- create_priorsense_data(drw)
```

example_powerscale_model *Example Stan model for power-scaling*

Description

Provides example models (with data) that are ready for use with power-scaling.

Usage

```
example_powerscale_model(model = "univariate_normal")
```

Arguments

model	Character specifying which model code to return. Currently "univariate_normal" and "eight_schools" are implemented.

Value

List containing model code and corresponding data.

Examples

```
ex_normal <- example_powerscale_model(model = "univariate_normal")

ex_eightschools <- example_powerscale_model(model = "eight_schools")
```

log_lik_draws *Extract log likelihood draws*

Description

Extract log likelihood from fitted model and return as a draws object.

Usage

```
log_lik_draws(x, ...)

## S3 method for class 'stanfit'
log_lik_draws(x, joint = FALSE, log_lik_name = "log_lik", ...)

## S3 method for class 'CmdStanFit'
log_lik_draws(x, joint = FALSE, log_lik_name = "log_lik", ...)

## S3 method for class 'draws'
log_lik_draws(x, joint = FALSE, log_lik_name = "log_lik", ...)
```

Arguments

x	Model fit or draws object.
...	Arguments passed to individual methods.
joint	Logical indicating whether to return the joint log likelihood or array. Default is FALSE.
log_lik_name	Name of parameter in Stan model corresponding to log likelihood, default is "log_lik".

Value

A draws_array object containing log_lik values.

Examples

```
ex <- example_powerscale_model()
drw <- ex$draws

log_lik_draws(drw)
```

---

log_prior_draws *Extract log prior draws*

Description

Extract log likelihood from fitted model and return as a draws object.

Usage

```
log_prior_draws(x, ...)

## S3 method for class 'stanfit'
log_prior_draws(x, joint = FALSE, log_prior_name = "lprior", ...)

## S3 method for class 'CmdStanFit'
log_prior_draws(x, joint = FALSE, log_prior_name = "lprior", ...)

## S3 method for class 'draws'
log_prior_draws(x, joint = FALSE, log_prior_name = "lprior", ...)
```

Arguments

x	Model fit or draws object.
...	Arguments passed to individual methods.
joint	Logical indicating whether to return the joint log prior or array. Default is FALSE.
log_prior_name	Name of parameter in Stan model corresponding to log prior, default is "lprior".

Value

A draws_array object containing log_prior values.

Examples

```
ex <- example_powerscale_model()
drw <- ex$draws

log_prior_draws(drw)
```

`powerscale-gradients` *Power-scale gradients*

Description

Calculate the numerical derivative of posterior quantities/divergence with respect to power-scaling the specified component (prior or likelihood). This is done using importance sampling (and optionally moment matching).

Usage

```
powerscale_gradients(x, ...)

## Default S3 method:
powerscale_gradients(x, ...)

## S3 method for class 'priorsense_data'
powerscale_gradients(
  x,
  variable = NULL,
  component = c("prior", "likelihood"),
  type = c("quantities", "divergence"),
  lower_alpha = 0.99,
  upper_alpha = 1.01,
  div_measure = "cjs_dist",
  measure_args = list(),
  moment_match = FALSE,
  k_threshold = 0.5,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  scale = FALSE,
  prior_selection = NULL,
  likelihood_selection = NULL,
  ...
)
```

Arguments

x	Model fit or draws object.
...	Further arguments passed to functions.
variable	Variables to compute sensitivity of. If NULL (default) sensitivity is computed for all variables.
component	Component to power-scale (prior or likelihood).
type	type of sensitivity to measure ("distance", "quantity"). Multiple options can be specified at the same time.
lower_alpha	lower power to scale component by, should be < 1 (default is 0.9).
upper_alpha	upper power to scale component by, should be > 1 (default is 1.1).
div_measure	The divergence measure to use. The following methods are implemented: • "cjs_dist": Cumulative Jensen-Shannon distance. Default method. See function cjs_dist for more details. • "js_dist": Jensen-Shannon distance. • "js_div": Jensen-Shannon divergence. • "hellinger_dist": Hellinger distance. • "kl_dist": Kullback-Leibler distance. • "kl_div": Kullback-Leibler divergence. • "ks_dist": Kolmogorov-Smirnov distance. • "hellinger_dist": Hellinger distance. • "ws_dist": Wassterstein distance (pass measure_args = list(p = N)) for a different order, where N is the order.
measure_args	Named list of further arguments passed to divergence measure functions.
moment_match	Logical; Indicate whether or not moment matching should be performed. Can only be TRUE if is_method is "psis".
k_threshold	Threshold value for Pareto k values above which the moment matching algorithm is used. Default is 0.5.
resample	Logical; Indicate whether or not draws should be resampled based on calculated importance weights.
transform	Indicate a transformation of posterior draws to perform before sensitivity analysis. Either "scale" or "whiten".
prediction	Function taking the model fit and returning a draws_df of predictions to be appended to the posterior draws
scale	logical scale quantity gradients by base posterior standard deviation.
prior_selection	Numeric vector specifying which priors to consider.
likelihood_selection	Numeric vector specifying which likelihoods to consider.

Value

Maximum of the absolute derivatives above and below alpha = 1.

Examples

```
ex <- example_powerscale_model()
drw <- ex$draws

powerscale_gradients(drw)
```

`powerscale-overview` *Prior/likelihood power-scaling perturbation*

Description

Estimate posterior draws based on power-scaling perturbations of prior or likelihood using importance sampling (and optionally moment matching).

Usage

```
powerscale(x, ...)

## Default S3 method:
powerscale(
  x,
  component,
  alpha,
  moment_match = FALSE,
  k_threshold = NULL,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  variable = NULL,
  selection = NULL,
  ...
)

## S3 method for class 'priorsense_data'
powerscale(
  x,
  component,
  alpha,
  moment_match = FALSE,
  k_threshold = NULL,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  variable = NULL,
  selection = NULL,
  ...
)
```

```
powerscale_sequence(x, ...)

## Default S3 method:
powerscale_sequence(
  x,
  lower_alpha = 0.8,
  upper_alpha = 1/lower_alpha,
  length = 3,
  variable = NULL,
  component = c("prior", "likelihood"),
  moment_match = FALSE,
  k_threshold = 0.5,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  auto_alpha_range = FALSE,
  symmetric = TRUE,
  prior_selection = NULL,
  likelihood_selection = NULL,
  ...
)

## S3 method for class 'priorsense_data'
powerscale_sequence(
  x,
  lower_alpha = 0.8,
  upper_alpha = 1/lower_alpha,
  length = 3,
  variable = NULL,
  component = c("prior", "likelihood"),
  moment_match = FALSE,
  k_threshold = 0.5,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  auto_alpha_range = FALSE,
  symmetric = TRUE,
  prior_selection = NULL,
  likelihood_selection = NULL,
  ...
)
```

Arguments

x	A fitted model object.
...	Further arguments passed to internal functions.
component	Component to be power-scaled (either "prior" or "likelihood"). For power-

scale_sequence, this can be both "prior" and "likelihood".

`alpha` Value by which to power-scale specified component. (likelihood/prior).

`moment_match` Logical; Indicate whether or not moment matching should be performed. Can only be TRUE if `is_method` is "psis".

`k_threshold` Threshold value for Pareto k values above which the moment matching algorithm is used. Default is 0.5.

`resample` Logical; Indicate whether or not draws should be resampled based on calculated importance weights.

`transform` Indicate a transformation of posterior draws to perform before sensitivity analysis. Either "scale" or "whiten".

`prediction` Function taking the model fit and returning a draws_df of predictions to be appended to the posterior draws

`variable` Vector of variable names to return estimated posterior draws for. If `NULL` all variables will be included.

`selection` Numeric vector specifying partitions of component to be included in power-scaling. Default is NULL, which takes all partitions.

`lower_alpha` Lower power-scaling alpha value in sequence.

`upper_alpha` Upper power-scaling alpha value in sequence.

`length` Length of alpha sequence.

`auto_alpha_range` Boolean. Restrict range to ensure Pareto-k values below threshold?

`symmetric` Boolean. Should the alpha range be symmetrical around alpha = 1, on log-space?

`prior_selection` Numeric vector of prior partitions to include in power-scaling. Default is NULL, which takes all partitions.

`likelihood_selection` Numeric vector of likelihood partitions to include in power-scaling. Default is NULL, which takes all partitions.

Value

A `powerscaled_draws` or `powerscaled_sequence` object, which contains the estimated posterior draws resulting from the power-scaling perturbations and details of the perturbation and estimation methods.

References

Kallioinen, N., Paananen, T., Bürkner, P-C., Vehtari, A. (2023). Detecting and diagnosing prior and likelihood sensitivity with power-scaling perturbations. *Statistics and Computing*. 34(57). `doi:10.1007/s11222-023-10366-5`

Vehtari, A., Simpson, D., Gelman, A., Yao, Y., and Gabry, J. (2024). Pareto smoothed importance sampling. *Journal of Machine Learning Research*. 25(72). `https://jmlr.org/papers/v25/19-556.html`

Paananen, T., Piironen, J., Bürkner, P-C., Vehtari, A. (2021). Implicitly adaptive importance sampling. *Statistics and Computing*. 31(16). `doi:10.1007/s11222-020-09982-2`

Examples

```
ex <- example_powerscale_model()

powerscale(ex$draws, component = "prior", alpha = 0.5)

powerscale_sequence(ex$draws)
```

`powerscale-sensitivity` *Power-scaling sensitivity analysis*

Description

Calculates the prior/likelihood sensitivity based on power-scaling perturbations. This is done using importance sampling (and optionally moment matching).

Usage

```
powerscale_sensitivity(x, ...)

## Default S3 method:
powerscale_sensitivity(
  x,
  variable = NULL,
  lower_alpha = 0.99,
  upper_alpha = 1.01,
  div_measure = "cjs_dist",
  measure_args = list(),
  component = c("prior", "likelihood"),
  sensitivity_threshold = 0.05,
  moment_match = FALSE,
  k_threshold = 0.5,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  prior_selection = NULL,
  likelihood_selection = NULL,
  num_args = NULL,
  ...
)

## S3 method for class 'priorsense_data'
powerscale_sensitivity(
  x,
  variable = NULL,
  lower_alpha = 0.99,
  upper_alpha = 1.01,
```

```
  div_measure = "cjs_dist",
  measure_args = list(),
  component = c("prior", "likelihood"),
  sensitivity_threshold = 0.05,
  moment_match = FALSE,
  k_threshold = 0.5,
  resample = FALSE,
  transform = NULL,
  prediction = NULL,
  prior_selection = NULL,
  likelihood_selection = NULL,
  num_args = NULL,
  ...
)

## S3 method for class 'CmdStanFit'
powerscale_sensitivity(x, ...)

## S3 method for class 'stanfit'
powerscale_sensitivity(x, ...)
```

Arguments

x	Model fit object or priorsense_data object.
...	Further arguments passed to functions.
variable	Character vector of variables to check.
lower_alpha	Lower alpha value for gradient calculation.
upper_alpha	Upper alpha value for gradient calculation.
div_measure	The divergence measure to use. The following methods are implemented: • "cjs_dist": Cumulative Jensen-Shannon distance. Default method. See function cjs_dist for more details. • "js_dist": Jensen-Shannon distance. • "js_div": Jensen-Shannon divergence. • "hellinger_dist": Hellinger distance. • "kl_dist": Kullback-Leibler distance. • "kl_div": Kullback-Leibler divergence. • "ks_dist": Kolmogorov-Smirnov distance. • "hellinger_dist": Hellinger distance. • "ws_dist": Wassterstein distance (pass measure_args = list(p = N)) for a different order, where N is the order.
measure_args	Named list of further arguments passed to divergence measure functions.
component	Character vector specifying component(s) to scale (default is both "prior" and "likelihood").
sensitivity_threshold	Threshold for flagging variable as sensitive to power-scaling.

moment_match	Logical; Indicate whether or not moment matching should be performed. Can only be TRUE if is_method is "psis".
k_threshold	Threshold value for Pareto k values above which the moment matching algorithm is used. Default is 0.5.
resample	Logical; Indicate whether or not draws should be resampled based on calculated importance weights.
transform	Indicate a transformation of posterior draws to perform before sensitivity analysis. Either "scale" or "whiten".
prediction	Function taking the model fit and returning a draws_df of predictions to be appended to the posterior draws
prior_selection	Numeric vector of prior partitions to include in power-scaling. Default is NULL, which takes all partitions.
likelihood_selection	Numeric vector of likelihood partitions to include in power-scaling. Default is NULL, which takes all partitions.
num_args	(named list) Optional arguments passed to num() for pretty printing of summaries. Can be controlled globally via the posterior.num_args option.

Value

Table of sensitivity values for each specified variable.

References

Kallioinen, N., Paananen, T., Bürkner, P-C., Vehtari, A. (2023). Detecting and diagnosing prior and likelihood sensitivity with power-scaling perturbations. *Statistics and Computing*. 34(57). doi:10.1007/s11222-023-10366-5

Vehtari, A., Simpson, D., Gelman, A., Yao, Y., and Gabry, J. (2024). Pareto smoothed importance sampling. *Journal of Machine Learning Research*. 25(72). https://jmlr.org/papers/v25/19-556.html

Paananen, T., Piironen, J., Bürkner, P-C., Vehtari, A. (2021). Implicitly adaptive importance sampling. *Statistics and Computing*. 31(16). doi:10.1007/s11222-020-09982-2

Examples

```
ex <- example_powerscale_model()
powerscale_sensitivity(ex$draws)
```

---

powerscale_derivative *Derivative with respect to power-scaling*

Description

Calculate the analytical derivative of a quantity with respect to power-scaling prior or likelihood.

Usage

```
powerscale_derivative(x, log_component, quantity = "mean", ...)
```

Arguments

x	Posterior draws.
log_component	Log likelihood or log prior values.
quantity	Character specifying quantity of interest (default is "mean"). Options are "mean", "sd", "var".
...	unused

Value

Derivative of the quantity with respect to log2 of the power-scaling factor (alpha).

Examples

```
example_model <- example_powerscale_model()
draws <- example_model$draws
log_prior <- log_prior_draws(draws, joint = TRUE)
posterior::summarise_draws(
    posterior::subset_draws(draws, variable = c("mu", "sigma")),
    mean,
    mean_sens = ~powerscale_derivative(.x, log_prior, quantity = "mean")
)
```

---

powerscale_plots *Diagnostic plots for power-scaling sensitivity*

Description

Various diagnostic plots for power-scaling sensitivity. See **Plot Descriptions** below for details.

Usage

```
powerscale_plot_dens(x, ...)

powerscale_plot_ecdf(x, ...)

## S3 method for class 'powerscaled_sequence'
powerscale_plot_ecdf(
  x,
  variable = NULL,
  resample = FALSE,
  length = 3,
  facet_rows = "component",
  help_text = getOption("priorsense.plot_help_text", TRUE),
```

```
  colors = NULL,
  ...
)

powerscale_plot_quantities(x, ...)

## S3 method for class 'powerscaled_sequence'
powerscale_plot_quantities(
  x,
  variable = NULL,
  quantity = c("mean", "sd"),
  div_measure = "cjs_dist",
  resample = FALSE,
  measure_args = NULL,
  mcse = TRUE,
  quantity_args = NULL,
  help_text = getOption("priorsense.plot_help_text", TRUE),
  colors = NULL,
  ...
)
```

Arguments

x	An object of class powerscaled_sequence or an object for which powerscale_sequence will first be run on.
...	Arguments passed to powerscale_sequence if x is not of class powerscaled_sequence.
variable	A character vector of variable names. If NULL (the default) all variables will be plotted.
resample	Logical; Indicate whether or not draws should be resampled based on calculated importance weights.
length	Numeric specifying how many alpha values should be used. Ignored of the object is of class powerscaled_sequence.
facet_rows	Character defining the rows of the plot facets, either "variable" or "component". Default is "variable".
help_text	Logical indicating whether title and subtitle with explanatory description should be included in the plot. Default is TRUE. Can be set via option "priorsense.show_help_text".
colors	Character vector of colors to be used for plots. Either length 3 for powerscale_plot_ecdf and powerscale_plot_dens with order lowest, base, highest; or length 2 for powerscale_plot_quantities with order low Pareto k, high Pareto k. If NULL the defaults will be used.
quantity	A character vector specifying one or several quantities to plot. Options are "mean", "median", "sd", "mad", "quantile".
div_measure	The divergence measure to use. The following methods are implemented: • "cjs_dist": Cumulative Jensen-Shannon distance. Default method. See function cjs_dist for more details. • "js_dist": Jensen-Shannon distance.

- "js_div": Jensen-Shannon divergence.
- "hellinger_dist": Hellinger distance.
- "kl_dist": Kullback-Leibler distance.
- "kl_div": Kullback-Leibler divergence.
- "ks_dist": Kolmogorov-Smirnov distance.
- "hellinger_dist": Hellinger distance.
- "ws_dist": Wassterstein distance (pass measure_args = list(p = N)) for a different order, where N is the order.

measure_args Named list of further arguments passed to divergence measure functions.

mcse Boolean; If TRUE will plot +/- 2 * Monte Carlo standard error of the base quantity on the quantities plot.

quantity_args Named list of further arguments passed to quantity functions. Passed as .args to [posterior::summarise_draws].

Value

A ggplot object that can be further customized using the **ggplot2** package.

Plot Descriptions

powerscale_plot_dens() Kernel density plot of power-scaled posterior draws with respect to power-scaling.

powerscale_plot_ecdf() Empirical cumulative distribution function plot of power-scaled posterior draws with respect to power-scaling.

powerscale_plot_quantities() Plot of posterior quantities with respect to power-scaling.

Examples

```
ex <- example_powerscale_model()

powerscale_plot_dens(ex$draws)
```

---

predictions_as_draws *brms predictions as draws*

Description

Create predictions using brms functions and convert them into draws format

Usage

```
predictions_as_draws(
  x,
  predict_fn,
  prediction_names = NULL,
  warn_dims = getOption("priorsense.warn", TRUE),
  ...
)
```

Arguments

x	brmsfit object
predict_fn	function for predictions
prediction_names	optional names of the predictions
warn_dims	throw a warning when coercing predict_fn's output from 3 margins to 2 margins?
...	further arguments passed to predict_fn

Value

draws array of predictions

Examples

```
## Not run:
library(brms)

if ("log_prior_draws.brmsfit" %in% methods(log_prior_draws) &&
    ("log_lik_draws.brmsfit" %in% methods(log_lik_draws))) {
  fit <- brm(
    yield ~ N * P * K,
    data = npk,
    prior = prior(normal(0, 1), class = "b"),
    refresh = 0
  )

  powerscale_sensitivity(
      fit,
      variable = "_pred",
      prediction = function(x) predictions_as_draws(
                                  x, brms::posterior_epred
                                )
  )
}

## End(Not run)
```

Index