Package ‘rater’

July 13, 2021

Title  Statistical Models of Repeated Categorical Rating Data

Version  1.2.0

Description  Fit statistical models based on the Dawid-Skene model - Dawid and Skene (1979) <doi:10.2307/2346806> - to repeated categorical rating data. Full Bayesian inference for these models is supported through the Stan modelling language. ‘rater’ also allows the user to extract and plot key parameters of these models.

License  GPL-2

URL  https://jeffreypullin.github.io/rater/,  
https://github.com/jeffreypullin/rater

BugReports  https://github.com/jeffreypullin/rater/issues

Depends  R (>= 3.4.0)

Imports  ggplot2 (>= 2.2.1), loo (> 2.0.0), methods, Rcpp (>= 0.12.0),  
RcppParallel (>= 5.0.1), rlang (> 0.2.0), rstan (>= 2.19.2),  
rstantools (>= 2.0.0)

Suggests  coda, covr, knitr, rmarkdown, testthat

LinkingTo  BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),  
RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

VignetteBuilder  knitr

Biarch  true

Config/testthat/edition  3

Encoding  UTF-8

LazyData  true

RoxygenNote  7.1.1

SystemRequirements  GNU make

NeedsCompilation  yes

Author  Jeffrey Pullin [aut, cre, cph]  
(<https://orcid.org/0000-0003-3651-5471>),  
Damjan Vukcevic [aut] (<https://orcid.org/0000-0001-7780-9586>),  
Lars Mølgaard Saxhaug [aut] (<https://orcid.org/0000-0001-5084-1578>)
**rater-package**

**Maintainer** Jeffrey Pullin &lt;jeffrey.pullin@gmail.com&gt;  
**Repository** CRAN  
**Date/Publication** 2021-07-13 14:40:02 UTC

### R topics documented:

- rater-package ................................................................. 2  
- anesthesia ................................................................. 3  
- as_mcmc.list ............................................................. 3  
- caries ........................................................................... 4  
- class_probabilities ...................................................... 5  
- get_stanfit ................................................................. 6  
- loo.rater_fit .............................................................. 6  
- mcmc_diagnostics ....................................................... 8  
- models ........................................................................... 9  
- plot.rater_fit .............................................................. 10  
- point_estimate ............................................................. 12  
- posterior_interval.mcmc_fit ......................................... 13  
- posterior_interval.optim_fit .......................................... 14  
- posterior_predict.rater_fit ............................................. 14  
- posterior_samples ....................................................... 15  
- print.mcmc_fit ............................................................ 16  
- print.optim_fit ............................................................ 17  
- print.rater_model ........................................................ 18  
- prior_summary.rater_fit ................................................ 18  
- rater ................................................................. 19  
- summary.mcmc_fit ....................................................... 20  
- summary.optim_fit ...................................................... 21  
- summary.rater_model ................................................... 22  
- waic.rater_fit ............................................................ 22  
- wide_to_long ............................................................... 23

### Index

- rater-package ......................................................... 25  

---

**rater-package**  
*The ‘rater’ package.*

### Description

Fit statistical models based on the Dawid-Skene model to repeated categorical rating data. Full Bayesian inference for these models is supported through the Stan modelling language. rater also allows the user to extract and plot key parameters of these models.

### References

http://mc-stan.org
**Description**

The data consist of ratings, on a 4-point scale, made by four anaesthetists of patients’ pre-operative health. The ratings were based on the anaesthetists assessments of a standard form completed for all of the patients. There are 45 patients (items) and four anaesthetists (raters) in total. The first anaesthetist assessed the forms a total of three times, spaced several weeks apart. The other anaesthetists each assessed the forms once. The data is in ‘long’ format.

**Usage**

anesthesia

**Format**

A data.frame with 315 rows and 3 columns:

- **item**: The item index - which item is being rated
- **rater**: The rater index - which rater is doing the rating
- **rating**: The rating given

**References**


---

**as_mcmc.list**

*Convert a rater_fit object to a coda mcmc.list object.*

**Description**

Convert a rater_fit object to a coda mcmc.list object.

**Usage**

as_mcmc.list(fit)

**Arguments**

- **fit**: A rater_fit object.

**Value**

A coda mcmc.list object.
Examples

# Fit a model using MCMC (the default).
mcmc_fit <- rater(anesthesia, "dawid_skene")

# Convert it to an mcmc.list
rater_mcmc_list <- as_mcmc.list(mcmc_fit)

---

caries

Dentist ratings of whether caries are healthy or not based on X-rays

Description

It consists of binary ratings, made by 5 dentists, of whether a given tooth was healthy (sound) or had caries, also known as cavities. The ratings were performed using X-ray only, which was thought to be more error-prone than visual/tactile assessment of each tooth. In total 3,689 ratings were made. This data is in 'grouped' format. Each row is one of the 'pattern' with the final columns being a tally of how many times that pattern occurs in the dataset.

Usage

caries

Format

A data.frame with 6 columns and 32 rows.

rater_1 The rating of the dentist 1
rater_2 The rating of the dentist 2
rater_3 The rating of the dentist 3
rater_4 The rating of the dentist 4
rater_5 The rating of the dentist 5
n The number of times the rating pattern appears in the dataset

References

class_probabilities

Extract latent class probabilities from a rater fit object

Description

Extract latent class probabilities from a rater fit object

Usage

```r
class_probabilities(fit, ...)  
## S3 method for class 'mcmc_fit'
class_probabilities(fit, ...)
## S3 method for class 'optim_fit'
class_probabilities(fit, ...)
```

Arguments

- `fit` A rater fit object.
- `...` Extra arguments.

Details

The latent class probabilities are obtained by marginalising out the latent class and then calculating, for each draw of pi and theta, the conditional probability of the latent class given the other parameters and the data. Averaging these conditional probabilities gives the (unconditional) latent class probabilities returned by this function.

Value

A I * K matrix where each element is the probably of item i being of class k. (I is the number of items and K the number of classes).

Examples

```r
fit <- rater(anesthesia, "dawid_skene")  
class_probabilities(fit)
```
get_stanfit

Get the underlying stanfit object from a rater_fit object.

Description

Get the underlying stanfit object from a rater_fit object.

Usage

get_stanfit(fit)

Arguments

fit
A rater_fit object.

Value

A stanfit object from rstan.

Examples

fit <- rater(anesthesia, "dawid_skene", verbose = FALSE)
stan_fit <- get_stanfit(fit)
stan_fit

loom.rater_fit

Compute the PSIS LOO CV - a measure of model fit - of a rater fit object.

Description

Compute the PSIS LOO CV - a measure of model fit - of a rater fit object.

Usage

## S3 method for class 'rater_fit'
loo(x, ..., cores = getOption("mc.cores", 1))
Arguments

- `x`: A `rater_fit` object. All model types are currently supported except the basic Dawid-Skene model fit with grouped data.
- `...`: Other arguments passed.
- `cores`: The number of cores to use when calling the underlying functions. By default the value of the `mc.cores` option.

Details

This function is somewhat experimental; model comparison is always difficult and choosing between variants of the Dawid-Skene model should be largely guided by considerations of data size and what is known about the characteristics of the raters. loo is, however, one of the leading methods for Bayesian model comparison and should provide a helpful guide in many situations.

When calculating loo we always use the relative effective sample size, calculated using `loo::relative_eff` to improve the estimates of the PSIS effective sample sizes and Monte Carlo error.

For further information about the details of loo and PSIS please consult the provided references.

Value

A loo object.

References


Examples

```r
fit_ds <- rater(anesthesia, "dawid_skene", verbose = FALSE, chains = 1)
fit_ccds <- rater(anesthesia, "class_conditional_dawid_skene",
                 verbose = FALSE, chains = 1)

loo_ds <- loo(fit_ds)
loo_ccds <- loo(fit_ccds)

# To compare the loos easily we can use the loo_compare function from the
# loo package:
library(loo)

loo_compare(loo_ds, loo_ccds)

# The documentation of the loo package contains more information about how
# the output should be interpreted.
```
mcmc_diagnostics

Retrieve MCMC convergence diagnostics for a rater fit

Description
Retrieve MCMC convergence diagnostics for a rater fit

Usage
mcmc_diagnostics(fit, pars = c("pi", "theta"))

Arguments
fit  An rater mcmc_fit object.
pars A character vector of parameter names to return. By default c("pi", "theta").

Details
MCMC diagnostics cannot be calculate for the z due to the marginalisation used to fit the models.
These MCMC diagnostics are intended as basic sanity check of the quality of the MCMC samples returned. Users who want more in depth diagnostics should consider using as_mcmc.list() to convert the samples to a coda::mcmc.list() object, or get_stanfit() to extract the underlying stanfit object.

Value
A matrix where the columns represent different diagnostics and the rows are different parameters. Currently the first column contains the Rhat statistic and the second bulk effective samples size. The rownames contain the parameter names.

References

See Also
rstan::Rhat(), rstan::ess_bulk() as_mcmc.list(), get_stanfit().
**Examples**

```r
fit <- rater(anesthesia, "dawid_skene")

# Calculate the diagnostics for all parameters.
mcmc_diagnostics(fit)

# Calculate the diagnostics just for the pi parameter.
mcmc_diagnostics(fit, pars = "pi")
```

---

**models**

**Probabilistic models of repeated categorical rating**

**Description**

Functions to set up models and change their prior parameters for use in `rater()`.

**Usage**

```r
dawid_skene(alpha = NULL, beta = NULL)

hier_dawid_skene(alpha = NULL)

class_conditional_dawid_skene(alpha = NULL, beta_1 = NULL, beta_2 = NULL)
```

**Arguments**

- `alpha`: prior parameter for pi
- `beta`: prior parameter for theta. This can either be a K * K matrix, in which case it is interpreted as the prior parameter of all of the J raters, or a J by K by K array in which case it is the fully specified prior parameter for all raters. (Here K is the number of categories in the data and J is the number of raters in the data.)
- `beta_1`: First on diagonal prior probability parameter
- `beta_2`: Second on diagonal prior probability parameter for theta

**Value**

A rater model object that can be passed to `rater()`.
Examples

# Model with default prior parameters:
default_m <- dawid_skene()

# Changing alpha:
set_alpha_m <- dawid_skene(alpha = c(2, 2, 2))

# Changing beta, single matrix:
# (See details for how this is interpreted.)
beta_mat <- matrix(1, nrow = 4, ncol = 4)
diag(betamat) <- 4
beta_mat_m <- dawid_skene()

# The above is equivalent (when the model is fit - see details) to:
beta_array <- array(NA, dim = c(2, 4, 4))
for (i in 1:2) {
  beta_array[i, , ] <- beta_mat
}
beta_array_m <- dawid_skene(beta = beta_array)

# But you can also specify an array where each slice is different.
# (Again, see details for how this is interpreted.)
beta_array[1, , ] <- matrix(1, nrow = 4, ncol = 4)
beta_array_m <- dawid_skene(beta = beta_array)

# Default:
hier_dawid_skene()

# Changing alpha
hier_dawid_skene(alpha = c(2, 2))

# Default:
class_conditional_dawid_skene()

# Not default:
class_conditional_dawid_skene(
  alpha = c(2, 2),
  beta_1 = c(4, 4),
  beta_2 = c(2, 2)
)

---

plot.rater_fit         Plot a rater_fit object

Description

Plot a rater_fit object
Usage

```r
## S3 method for class 'rater_fit'
plot(x, pars = "theta", prob = 0.9, rater_index = NULL, item_index = NULL, ...)
```

Arguments

- `x`: An object of class `rater_fit`.
- `pars`: A length one character vector specifying the parameter to plot. By default "theta".
- `prob`: The coverage of the credible intervals shown in the "pi" plot. If not plotting pi this argument will be ignored. By default 0.9.
- `rater_index`: The indexes of the raters shown in the "theta plot. If not plotting theta this argument will be ignored. By default NULL which means that all raters will be plotted.
- `item_index`: The indexes of the items shown in the class probabilities plot. If not plotting the class probabilities this argument will be ignored. By default NULL which means that all items will be plotted. This argument is particularly useful to focus the subset of items with substantial uncertainty in their class assignments.
- `...`: Other arguments.

Details

The use of `pars` to refer to only one parameter is for backwards compatibility and consistency with the rest of the interface.

Value

A ggplot2 object.

Examples

```r
fit <- rater(anesthesia, "dawid_skene")
# By default will just plot the theta plot
plot(fit)

# Select which parameter to plot.
plot(fit, pars = "pi")
```
point_estimate

Extract point estimates of parameters from a fit object

Description

Extract point estimates of parameters from a fit object

Usage

point_estimate(fit, pars = c("pi", "theta", "z"), ...)

Arguments

| fit       | A rater fit object |
| pars      | A character vector of parameter names to return. By default c("pi", "theta", "z"). |
| ...       | Extra arguments |

Details

If the passed fit object was fit using MCMC then the posterior means are returned. If it was fit through optimisation the maximum a priori (MAP) estimates are returned. The z parameter returned is the value of class probabilities which is largest. To return the full posterior distributions of the latent class use class_probabilities().

For the class conditional model the 'full' theta parameterisation (i.e. appearing to have the same number of parameters as the standard Dawid-Skene model) is calculated and returned. This is designed to allow easier comparison with the full Dawid-Skene model.

Value

A named list of the parameter estimates.

See Also

class_probabilities()

Examples

# A model fit using MCMC.
mcmc_fit <- rater(anesthesia, "dawid_skene")

# This will return the posterior mean (except for z)
post_mean_estimate <- point_estimate(mcmc_fit)

# A model fit using optimisation.
optim_fit <- rater(anesthesia, dawid_skene(), method = "optim")
# This will output MAP estimates of the parameters.
map_estimate <- point_estimate(optim_fit)

## S3 method for class 'mcmc_fit'
posterior_interval(object, prob = 0.9, pars = c("pi", "theta"), ...)

Arguments

- `object` A rater `mcmc_fit` object.
- `prob` A single probability. The size of the credible interval returned. By default 0.9.
- `pars` The parameters to calculate the intervals for
- `...` Other arguments.

Details

Posterior intervals can only be calculated for models fit with MCMC. In addition, posterior intervals are not meaningful for the latent class (and indeed cannot be calculated). The full posterior distribution of the latent class can be extracted using `class_probabilities`.

For the class conditional model the 'full' theta parameterisation (i.e. appearing to have the same number of parameters as the standard Dawid-Skene model) is calculated and returned. This is designed to allow easier comparison with the full Dawid-Skene model.

Value

A matrix with 2 columns. The first column is the lower bound of the credible interval and the second is the upper bound. Each row corresponds to one individual's parameters. The rownames are the parameter names.
Examples

```r
fit <- rater(anesthesia, "dawid_skene", verbose = FALSE, chains = 1)
intervals <- posterior_interval(fit)
head(intervals)
```

---

### posterior_interval.optim_fit

Extract posterior intervals for parameters of the model

**Description**

Extract posterior intervals for parameters of the model

**Usage**

```r
## S3 method for class 'optim_fit'
posterior_interval(object, prob = 0.9, pars = c("pi", "theta"), ...)
```

**Arguments**

- `object` A rater optim_fit object
- `prob` A probability
- `pars` The parameters to calculate the intervals for
- `...` Other arguments

---

### posterior_predict.rater_fit

Draw from the posterior predictive distribution

**Description**

Draw from the posterior predictive distribution

**Usage**

```r
## S3 method for class 'rater_fit'
posterior_predict(object, new_data, seed = NULL, ...)
```
posterior_samples

Arguments

object A rater_fit object.
new_data New data for the model to be fit to. The must be in the form used in rater() except without the 'rating' column.
seed An optional random seed to use.
... Other arguments.

Details

The number of raters implied by the entries in the rater column must match the number of raters in the fitted model.
Due to technical issues drawing from the posterior predictive distribution of the hierarchical Dawid-Skene model is currently not supported.

Value

The passed new_data augmented with a column 'z' containing the latent class of each item and 'rating' containing the simulated rating.

Examples

fit <- rater(anesthesia, "dawid_skene", verbose = FALSE)
new_data <- data.frame(item = rep(1:2, each = 5), rater = rep(1:5, 2))
predictions <- posterior_predict(fit, new_data)
predictions

posterior_samples Extract posterior samples from a rater fit object

Description

Extract posterior samples from a rater fit object

Usage

posterior_samples(fit, pars = c("pi", "theta"))

Arguments

fit A rater fit object.
pars A character vector of parameter names to return. By default c("pi", "theta").
Details

Posterior samples can only be returned for models fitting using MCMC not optimisation. In addi-
tion, posterior samples cannot be returned for the latent class due to the marginalisation technique
used internally.

For the class conditional model the 'full' theta parameterisation (i.e. appearing to have the same
number of parameters as the standard Dawid-Skene model) is calculated and returned. This is
designed to allow easier comparison with the full Dawid-Skene model.

Value

A named list of the posterior samples for each parameters. For each parameter the samples are in
the form returned by rstan::extract().

Examples

fit <- rater(anesthesia, "dawid_skene")
samples <- posterior_samples(fit)

# Look at first 6 samples for each of the pi parameters
head(samples$pi)

# Look at the first 6 samples for the theta[1, 1, 1] parameter
head(samples$theta[, 1, 1, 1])

# Only get the samples for the pi parameter:
pi_samples <- posterior_samples(fit, pars = "pi")
print.optim_fit

Examples

# Suppress sampling output.
mcmc_fit <- rater(anesthesia, "dawid_skene", verbose = FALSE)
print(mcmc_fit)

print.optim_fit

Print an optim_fit object

Description

Print an optim_fit object

Usage

## S3 method for class 'optim_fit'
print(x, 

Arguments

x

An object of class optim_fit.

... Other arguments.

Examples

optim_fit <- rater(anesthesia, "dawid_skene", method = "optim")
print(optim_fit)
print.rater_model   \textit{Print a rater\_model object.}

\section*{Description}
Print a rater\_model object.

\section*{Usage}
\begin{verbatim}
## S3 method for class 'rater_model'
print(x, ...)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
\item \textbf{x} \quad \text{A rater\_model object.}
\item \textbf{...} \quad \text{Other arguments}
\end{itemize}

\section*{Examples}
\begin{verbatim}
mod <- dawid_skene()
print(mod)
\end{verbatim}

\section*{prior_summary.rater_fit}
\textit{Provide a summary of the priors specified in a rater\_fit object.}

\section*{Description}
Provide a summary of the priors specified in a rater\_fit object.

\section*{Usage}
\begin{verbatim}
## S3 method for class 'rater_fit'
prior_summary(object, ...)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
\item \textbf{object} \quad \text{A rater\_fit object.}
\item \textbf{...} \quad \text{Other arguments.}
\end{itemize}
Examples

```r
# Fit a model using MCMC (the default).
fit <- rater(anesthesia, "dawid_skene", verbose = FALSE)

# Summarise the priors (and model) specified in the fit.
prior_summary(fit)
```

rater

Fit statistical models to repeated categorical rating data using Stan

Description

This function allows the user to fit statistical models of noisy categorical rating, based on the Dawid-Skene model, using Bayesian inference. A variety of data formats and models are supported. Inference is done using Stan, allowing models to be fit efficiently, using both optimisation and Markov Chain Monte Carlo (MCMC).

Usage

```r
rater(
  data, model,
  method = "mcmc",
  data_format = "long",
  inits = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

- **data**: A 2D data object: data.frame, matrix, tibble etc. with data in either long or grouped format.
- **model**: Model to fit to data - must be rater_model or a character string - the name of the model. If the character string is used, the prior parameters will be set to their default values.
- **method**: A length 1 character vector, either "mcmc" or "optim". This will be fitting method used by Stan. By default "mcmc"
- **data_format**: A length 1 character vector, "long", "wide" and "grouped". The format that the passed data is in. Defaults to "long". See vignette("data-formats) for details.
- **inits**: The initialization points of the fitting algorithm
- **verbose**: Should rater() produce information about the progress of the chains while using the MCMC algorithm. Defaults to TRUE
- **...**: Extra parameters which are passed to the Stan fitting interface.
summary.mcmc_fit

Details
The default MCMC algorithm used by Stan is No U Turn Sampling (NUTS) and the default optimisation method is LGFGS. For MCMC 4 chains are run by default with 2000 iterations in total each.

Value
An object of class rater_fit containing the fitted parameters.

See Also
rstan::sampling(), rstan::optimizing()

Examples

# Fit a model using MCMC (the default).
mcmc_fit <- rater(anesthesia, "dawid_skene")

# Fit a model using optimisation.
optim_fit <- rater(anesthesia, dawid_skene(), method = "optim")

# Fit a model using passing data grouped data.
grouped_fit <- rater(caries, dawid_skene(), data_format = "grouped")

summary.mcmc_fit  Summarise a mcmc_fit object

Description
Summarise a mcmc_fit object

Usage
## S3 method for class 'mcmc_fit'
summary(object, n_pars = 8, ...)

Arguments

object  An object of class mcmc_fit.
n_pars  The number of pi/theta parameters and z ‘items’ to display.
...  Other arguments passed to function.
Details

For the class conditional model the 'full' theta parameterisation (i.e. appearing to have the same number of parameters as the standard Dawid-Skene model) is calculated and returned. This is designed to allow easier comparison with the full Dawid-Skene model.

Examples

```r
fit <- rater(anesthesia, "dawid_skene", verbose = FALSE)
summary(fit)
```

### Summary.optim_fit

#### Summarise an optim_fit object

**Description**

Summarise an optim_fit object

**Usage**

```r
## S3 method for class 'optim_fit'
summary(object, n_pars = 8, ...)
```

**Arguments**

- **object**
  - An object of class optim_fit.
- **n_pars**
  - The number of pi/theta parameters and z 'items' to display.
- **...**
  - Other arguments passed to function.

**Details**

For the class conditional model the 'full' theta parameterisation (i.e. appearing to have the same number of parameters as the standard Dawid-Skene model) is calculated and returned. This is designed to allow easier comparison with the full Dawid-Skene model.

**Examples**

```r
fit <- rater(anesthesia, "dawid_skene", method = "optim")
summary(fit)
```
### Summary.rater_model

**Summarise a rater_model.**

**Description**

Summarise a rater_model.

**Usage**

```r
## S3 method for class 'rater_model'
summary(object, ...)
```

**Arguments**

- `object`: A rater_model object.
- `...`: Other arguments.

**Examples**

```r
mod <- dawid_skene()
summary(mod)
```

### Waic.rater_fit

**Compute the WAIC - a measure of model fit - of a rater fit object.**

**Description**

Compute the WAIC - a measure of model fit - of a rater fit object.

**Usage**

```r
## S3 method for class 'rater_fit'
waic(x, ...)
```

**Arguments**

- `x`: A rater_fit object. All model types are currently supported except the basic Dawid-Skene model fit with grouped data.
- `...`: Other arguments passed.
Details

This function provides an additional method for model comparison, on top of the `loo()` function. In general we recommend that `loo()` is preferred: see the documentation of the loo package for details. Also, note the comments regarding model selection in the details section of `loo()`.

Value

A waic/loo object.

References


Examples

```r
fit_ds <- rater(anesthesia, "dawid_skene", verbose = FALSE, chains = 1)
fit_ccds <- rater(anesthesia, "class_conditional_dawid_skene",
                   verbose = FALSE, chains = 1)
waic(fit_ds)
waic(fit_ccds)
```

---

**wide_to_long**

Convert wide data to the long format

**Description**

Convert wide data to the long format

**Usage**

`wide_to_long(data)`

**Arguments**

- `data` Data in a wide format. Must be 2D data object which can be converted to a data.frame
Details
Wide data refers to a way of laying out categorical rating data where each item is one row and each column represents the ratings of each rater. Elements of the data can be NA, indicating that an item wasn’t rated by a rater. Wide data cannot represent the same rater rating an item multiple times. Currently any column names of the data are ignored and the raters are labelled by their column position (1 indexed, left to right). Only numeric ratings are currently supported.

Value
The data converted into long format. A data.frame with three columns item, rater and rating.

Examples
```r
wide_data <- data.frame(dater_1 = c(3, 2, 2), rater_2 = c(4, 2, 2))
wide_data

long_data <- wide_to_long(wide_data)
long_data
```
Index

* datasets
  anesthesia, 3
caries, 4

anesthesia, 3
as_mcmc.list, 3
as_mcmc.list(), 8
caries, 4
class_conditional_dawid_skene (models), 9
class_probabilities, 5, 13
coda::mcmc.list, 8
dawid_skene (models), 9
get_stanfit, 6
get_stanfit(), 8
hier_dawid_skene (models), 9

loo (loo.rater_fit), 6
loo.rater_fit, 6
mcmc_diagnostics, 8
models, 9

plot.rater_fit, 10
point_estimate, 12
posterior_interval
  (posterior_interval.mcmc_fit), 13
posterior_interval.mcmc_fit, 13
posterior_interval.optim_fit, 14
posterior_predict
  (posterior_predict.rater_fit), 14
posterior_predict.rater_fit, 14
posterior_samples, 15
print.mcmc_fit, 16
print.optim_fit, 17

print.rater_model, 18
prior_summary
  (prior_summary.rater_fit), 18
prior_summary.rater_fit, 18
rater, 19
rater(), 9
rater-package, 2
rstan::ess_bulk, 8
rstan::extract, 16
rstan::optimizing, 20
rstan::Rhat(), 8
rstan::sampling, 20
summary.mcmc_fit, 20
summary.optim_fit, 21
summary.rater_model, 22

waic (waic.rater_fit), 22
waic.rater_fit, 22
wide_to_long, 23