Package: clmstan (via r-universe)

May 13, 2026

Type Package

Title Cumulative Link Models with 'CmdStanR'

Version 0.1.1

Author Tomotaka Momozaki [aut, cre]

Maintainer Tomotaka Momozaki <momozaki.stat@gmail.com>

Description Fits cumulative link models (CLMs) for ordinal categorical data using 'CmdStanR'. Supports various link functions including logit, probit, cloglog, loglog, cauchit, and flexible parametric links such as Generalized Extreme Value (GEV), Asymmetric Exponential Power (AEP), and Symmetric Power. Models are pre-compiled using the 'instantiate' package for fast execution without runtime compilation. Methods are described in Agresti (2010, ISBN:978-0-470-08289-8), Wang and Dey (2011) <doi:10.1007/s10651-010-0154-8>, and Naranjo, Perez, and Martin (2015) <doi:10.1007/s11222-014-9449-1>.

License MIT + file LICENSE

URL https://t-momozaki.github.io/clmstan/, https://github.com/t-momozaki/clmstan

BugReports https://github.com/t-momozaki/clmstan/issues

Depends R (>= 4.0.0)

Imports instantiate, posterior, bayesplot, loo, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ordinal, cmdstanr (>= 0.9.0)

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository https://t-momozaki.r-universe.dev

Date/Publication 2026-02-11 23:42:33 UTC

RemoteUrl https://github.com/t-momozaki/clmstan

RemoteRef HEAD

RemoteSha 81ff4f2a2e74d3c47401cedd603587f49083c235

Contents

c.clm_prior_spec *Combine Multiple Prior Specifications*

Description

Combines multiple `prior()` objects into a single prior specification list.

Usage

```
## S3 method for class 'clm_prior_spec'
c(...)
```

Arguments

`...`	`prior()` objects to combine.

Value

An object of class `"clm_prior_list"` containing all prior specifications.

Examples

```
# Combine multiple priors
priors <- c(
  prior(normal(0, 2.5), class = "b"),
  prior(normal(0, 10), class = "Intercept"),
  prior(gamma(2, 0.1), class = "df")
)
print(priors)
```

cauchy *Cauchy Distribution for Prior Specification*

Description

Creates a Cauchy distribution object for use with `prior()`.

Usage

```
cauchy(mu = 0, sigma = 1)
```

Arguments

`mu`	Location parameter. Default: 0
`sigma`	Scale parameter. Must be positive. Default: 1

Value

An object of class `"clm_dist"` representing a Cauchy distribution.

See Also

`prior()`, `normal()`, `gamma()`, `student_t()`

Examples

```
# Create a Cauchy prior (weakly informative)
cauchy(0, 2.5)

# Use with prior()
prior(cauchy(0, 2.5), class = "b")
```

`clm_prior` *Prior Specification for clmstan*

Description

Create prior specifications for cumulative link models in clmstan.

Default priors:

- Regression coefficients (beta): `normal(0, 2.5)`
- Cutpoints (c): `normal(0, 10)` for flexible, `normal(0, 5)` for symmetric
- Interval (d): `gamma(2, 0.5)` for equidistant threshold

Link parameter priors (when estimated):

Link	Parameter	Default Prior
tlink	df	gamma(2, 0.1)
aranda_ordaz	lambda	gamma(0.5, 0.5)
gev	xi	normal(0, 2)
sp	r	gamma(0.5, 0.5)
log_gamma	lambda	normal(0, 1)
aep	theta1, theta2	gamma(2, 1)

Usage

```
clm_prior(
  beta_sd = NULL,
  c_sd = NULL,
  c1_mu = NULL,
  c1_sd = NULL,
  d_alpha = NULL,
  d_beta = NULL,
```

```
  cpos_sd = NULL,
  df_alpha = NULL,
  df_beta = NULL,
  lambda_ao_alpha = NULL,
  lambda_ao_beta = NULL,
  lambda_lg_mu = NULL,
  lambda_lg_sd = NULL,
  xi_mu = NULL,
  xi_sd = NULL,
  r_alpha = NULL,
  r_beta = NULL,
  theta1_alpha = NULL,
  theta1_beta = NULL,
  theta2_alpha = NULL,
  theta2_beta = NULL
)
```

Arguments

beta_sd	SD for normal prior on regression coefficients. Default: 2.5 (weakly informative)
c_sd	SD for normal prior on cutpoints (flexible threshold). Default: 10
c1_mu	Mean for normal prior on first cutpoint (equidistant threshold). Default: 0
c1_sd	SD for normal prior on first cutpoint (equidistant threshold). Default: 10
d_alpha	Gamma shape for interval d (equidistant threshold). Default: 2
d_beta	Gamma rate for interval d (equidistant threshold). Default: 0.5
cpos_sd	SD for half-normal prior on positive cutpoints (symmetric threshold). Default: 5
df_alpha	Gamma shape for tlink df. Default: 2
df_beta	Gamma rate for tlink df. Default: 0.1
lambda_ao_alpha	Gamma shape for aranda_ordaz lambda. Default: 0.5
lambda_ao_beta	Gamma rate for aranda_ordaz lambda. Default: 0.5
lambda_lg_mu	Normal mean for log_gamma lambda. Default: 0
lambda_lg_sd	Normal SD for log_gamma lambda. Default: 1
xi_mu	Normal mean for GEV xi. Default: 0
xi_sd	Normal SD for GEV xi. Default: 2
r_alpha	Gamma shape for SP r. Default: 0.5
r_beta	Gamma rate for SP r. Default: 0.5
theta1_alpha	Gamma shape for AEP theta1. Default: 2
theta1_beta	Gamma rate for AEP theta1. Default: 1
theta2_alpha	Gamma shape for AEP theta2. Default: 2
theta2_beta	Gamma rate for AEP theta2. Default: 1

Value

An object of class "clm_prior" containing prior specifications.

Examples

```
# Create a prior object (does not require Stan)
my_prior <- clm_prior(beta_sd = 2, c_sd = 5)
print(my_prior)

## Not run:
# Examples below require CmdStan and compiled Stan models
data(wine, package = "ordinal")

# Default priors (no customization needed)
fit <- clm_stan(rating ~ temp, data = wine,
                chains = 2, iter = 500, warmup = 250, refresh = 0)

# Custom prior for regression coefficients
fit2 <- clm_stan(rating ~ temp, data = wine,
                 prior = clm_prior(beta_sd = 1),
                 chains = 2, iter = 500, warmup = 250, refresh = 0)

## End(Not run)
```

---

clm_stan *Fit a Cumulative Link Model using CmdStanR*

Description

Fit a Cumulative Link Model using CmdStanR

Usage

```
clm_stan(
  formula,
  data,
  link = "logit",
  base = "logit",
  threshold = "flexible",
  link_param = NULL,
  prior = NULL,
  chains = 4,
  iter = 2000,
  warmup = NULL,
  ...
)
```

Arguments

formula	A formula specifying the model (response ~ predictors)
data	A data frame containing the variables in the formula
link	Link function. One of "logit" (default), "probit", "cloglog", "loglog", "cauchit", "tlink", "gev", "aep", "sp", "aranda_ordaz", "log_gamma"
base	Base distribution for SP link. One of "logit" (default), "probit", "cloglog", "loglog", "cauchit", "tlink". Ignored for other link functions.
threshold	Threshold structure. One of "flexible" (default), "equidistant", "symmetric"
link_param	A list of link parameters. For flexible links, values can be: • Numeric: Use as fixed value (e.g., `list(df = 8)`) • "estimate": Estimate the parameter with Bayesian inference
prior	Prior specification. Can be either: • A `clm_prior` object created by `clm_prior()` • A distribution-based prior using `prior()` with distribution functions (`normal()`, `gamma()`, `student_t()`, `cauchy()`)
chains	Number of MCMC chains (default: 4)
iter	Total iterations per chain (default: 2000)
warmup	Warmup iterations per chain. If NULL (default), uses floor(iter/2)
...	Additional arguments passed to cmdstanr::sample()

Value

An object of class "clmstan"

Examples

```
## Not run:
# Fit a proportional odds model
library(ordinal)
data(wine)
fit <- clm_stan(rating ~ temp + contact, data = wine, link = "logit")
print(fit)

# Fit with t-link (fixed df)
fit_t <- clm_stan(rating ~ temp, data = wine, link = "tlink",
                  link_param = list(df = 8))

# Fit with GEV link (estimate xi)
fit_gev <- clm_stan(rating ~ temp, data = wine, link = "gev",
                    link_param = list(xi = "estimate"))

## End(Not run)
```

clmstan-class *clmstan S3 Class*

Description

The `clmstan` class represents a fitted cumulative link model. It contains the CmdStanR fit object and additional metadata.

Slots

fit The CmdStanMCMC object from cmdstanr

formula The model formula

data The original data frame

link The link function used

base The base distribution (for SP link)

threshold The threshold structure

link_param Link parameter settings (for flexible links)

full TRUE if link parameters were estimated (Bayesian inference), FALSE if they were fixed at user-specified values

K Number of response categories (cached from data)

N Number of observations (extracted from data for efficiency)

P Number of predictors (extracted from design matrix)

coef.clmstan *Extract coefficients from clmstan objects*

Description

Returns posterior point estimates (mean or median) for all model parameters.

Usage

```
## S3 method for class 'clmstan'
coef(object, type = c("mean", "median"), ...)
```

Arguments

`object`	A clmstan object
`type`	Type of point estimate: "mean" (default) or "median"
`...`	Additional arguments (ignored)

Value

A named numeric vector with:

- Threshold coefficients (e.g., "1|2", "2|3", ...)
- Regression coefficients (variable names from formula)

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
coef(fit)
coef(fit, type = "median")

## End(Not run)
```

`diagnostics` *MCMC Diagnostics for clmstan objects*

Description

Provides a summary of MCMC convergence diagnostics including HMC-specific diagnostics (divergences, treedepth, E-BFMI) and general convergence measures (Rhat, ESS).

Usage

```
diagnostics(object, ...)

## S3 method for class 'clmstan'
diagnostics(
  object,
  detail = FALSE,
  rhat_threshold = 1.01,
  ess_threshold = 400,
  ...
)
```

Arguments

`object`	A clmstan object
`...`	Additional arguments (ignored)
`detail`	Logical. If TRUE, show full parameter-level diagnostics table. If FALSE (default), show only summary and any problematic parameters.
`rhat_threshold`	Threshold for flagging high Rhat values. Default 1.01.
`ess_threshold`	Threshold for flagging low ESS values. Default 400.

Details

The function checks for the following issues:

- **Divergences**: Number of divergent transitions (ideally 0)
- **Treedepth**: Transitions hitting max treedepth (efficiency issue)
- **E-BFMI**: Energy Bayesian Fraction of Missing Information (values < 0.3 indicate problems)
- **Rhat**: Potential scale reduction factor (values > 1.01 indicate lack of convergence)
- **ESS**: Effective sample size for bulk and tail (low values indicate high autocorrelation)

Value

Invisibly returns a list containing:

- hmc: HMC diagnostics from CmdStanMCMC$diagnostic_summary()
- convergence: Data frame of per-parameter Rhat and ESS values
- issues: Logical indicating whether any issues were detected

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
diagnostics(fit)
diagnostics(fit, detail = TRUE)

## End(Not run)
```

---

extract_acf *Extract ACF values from clmstan object*

Description

Computes autocorrelation function (ACF) values for MCMC chains and returns them in a tidy data frame format.

Usage

```
extract_acf(object, pars = NULL, lags = 20, ...)
```

Arguments

object	A clmstan object
pars	Character vector of parameter names. If NULL (default), uses beta, c_transformed (except first), and beta0.
lags	Maximum number of lags to compute. Default is 20.
...	Additional arguments (ignored)

Details

The ACF measures how correlated each draw is with previous draws in the same chain. High autocorrelation at many lags indicates slow mixing and the need for more samples or reparameterization.

Ideally, ACF should drop to near zero within a few lags. Persistent high autocorrelation suggests the sampler is exploring the posterior slowly.

Value

A data frame with columns:

- parameter: Parameter name
- chain: Chain number
- lag: Lag value (0, 1, 2, ...)
- acf: Autocorrelation value

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
acf_df <- extract_acf(fit)
head(acf_df)

# Plot ACF for specific parameters
library(ggplot2)
acf_df |>
  dplyr::filter(parameter == "beta[1]") |>
  ggplot(aes(x = lag, y = acf, color = factor(chain))) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = "dashed")

## End(Not run)
```

`fitted.clmstan` *Fitted values for clmstan objects*

Description

Returns expected category probabilities for each observation. This is equivalent to `predict(object, type = "probs", summary = TRUE)`.

Usage

```
## S3 method for class 'clmstan'
fitted(
  object,
  newdata = NULL,
```

```
  summary = TRUE,
  robust = FALSE,
  probs = c(0.025, 0.975),
  ndraws = NULL,
  ...
)
```

Arguments

object	A clmstan object returned by clm_stan().
newdata	Optional data frame for prediction. If NULL (default), predictions are made for the original training data.
summary	Logical. If TRUE (default), return summary statistics (mean, SD, quantiles). If FALSE, return raw posterior draws.
robust	Logical. If TRUE, use median instead of mean for point estimates. Default is FALSE.
probs	Numeric vector of probabilities for quantiles. Default is c(0.025, 0.975) for 95% credible intervals.
ndraws	Number of posterior draws to use. If NULL (default), all available draws are used.
...	Additional arguments (currently ignored).

Value

If summary = TRUE (default): A data frame with N rows and columns for each category probability (P[Y=1], P[Y=2], etc.). If summary = FALSE: An S x N x K array of probability draws.

See Also

predict.clmstan(), posterior_predict.clmstan()

---

flat *Flat (Improper Uniform) Prior Distribution*

Description

Creates a flat (improper uniform) distribution object for use with prior(). A flat prior assigns equal probability density to all values, which is improper (does not integrate to 1) but can be used when the likelihood provides sufficient information for identification.

Usage

```
flat()
```

Value

An object of class "clm_dist" representing a flat distribution.

Note

Flat priors are supported for:

- Regression coefficients (class "b")
- Threshold classes ("Intercept", "c1", "cpos")

Using flat priors may lead to improper posteriors if the likelihood does not provide sufficient information. For thresholds with ordered constraints, Stan's internal transformation provides implicit regularization.

See Also

`prior()`, `normal()`, `student_t()`, `cauchy()`

Examples

```
# Create a flat prior for regression coefficients
prior(flat(), class = "b")

# Flat prior for thresholds (flexible)
prior(flat(), class = "Intercept")
```

---

gamma *Gamma Distribution for Prior Specification*

Description

Creates a gamma distribution object for use with `prior()`.

Usage

```
gamma(alpha, beta)
```

Arguments

`alpha`	Shape parameter of the gamma distribution. Must be positive.
`beta`	Rate parameter of the gamma distribution. Must be positive.

Value

An object of class `"clm_dist"` representing a gamma distribution.

Note

This function masks `base::gamma()`. To use the base gamma function, use `base::gamma()` explicitly.

See Also

prior(), normal(), student_t(), cauchy()

Examples

```
# Create a gamma prior
gamma(2, 0.1)

# Use with prior() for degrees of freedom
prior(gamma(2, 0.1), class = "df")
```

---

is.clmstan *Check if object is clmstan*

Description

Check if object is clmstan

Usage

```
is.clmstan(x)
```

Arguments

x	An object to test

Value

TRUE if x is a clmstan object

---

link_functions *Available Link Functions*

Description

clmstan supports the following link functions for cumulative link models:

Standard links (no additional parameters):

- "logit" - Logistic (proportional odds model)
- "probit" - Normal (latent variable interpretation)
- "cloglog" - Complementary log-log (proportional hazards)
- "loglog" - Log-log (Gumbel minimum)
- "cauchit" - Cauchy (heavy tails)

Flexible links (with additional parameters):

- "tlink" - Student-t (df > 0)
  - df = Inf: equals probit
  - df < 3: increasingly heavy tails; df > 30 is nearly normal
- "aranda_ordaz" - Aranda-Ordaz asymmetric (lambda > 0)
  - lambda = 1: equals logit
  - lambda -> 0: approaches cloglog
- "gev" - Generalized extreme value (shape parameter xi)
  - xi = 0: Gumbel (equals loglog)
  - xi < 0: Weibull (short tail)
  - xi > 0: Frechet (heavy tail)
- "sp" - Symmetric power (r > 0, base distribution)
  - r = 1: equals base distribution
  - 0 < r < 1: positively skewed
  - r > 1: negatively skewed
- "log_gamma" - Log-gamma (lambda)
  - lambda = 0: equals probit
  - lambda > 0 or < 0: asymmetric
- "aep" - Asymmetric exponential power (theta1 > 0, theta2 > 0)
  - alpha = 0.5 fixed for identifiability
  - theta1 = theta2: symmetric distribution
  - theta = 2: Gaussian kernel (but NOT equal to probit due to scaling)
  - theta < 2: heavy tails (leptokurtic)
  - theta > 2: light tails (platykurtic)

Link Parameter Specification

Flexible link parameters can be either **fixed** or **estimated** (inferred).

Fixed parameters: Specify a numeric value

```
clm_stan(y ~ x, link = "tlink", link_param = list(df = 8))
clm_stan(y ~ x, link = "gev", link_param = list(xi = 0))  # equals loglog
clm_stan(y ~ x, link = "aep", link_param = list(theta1 = 2, theta2 = 2)) # symmetric
```

Estimated parameters: Use "estimate" (with default prior)

```
clm_stan(y ~ x, link = "tlink", link_param = list(df = "estimate"))
clm_stan(y ~ x, link = "gev", link_param = list(xi = "estimate"))
```

Custom priors: Combine "estimate" with prior argument

```
clm_stan(y ~ x, link = "gev",
         link_param = list(xi = "estimate"),
         prior = prior(normal(0, 0.3), class = "xi"))
```

Default Priors for Link Parameters

When using `"estimate"`, the following default priors are used:

Link	Parameter	Default Prior	Notes
tlink	df	gamma(2, 0.1)	Mode around 10, allows heavy tails
aranda_ordaz	lambda	gamma(0.5, 0.5)	Centered near 1 (logit)
gev	xi	normal(0, 2)	Weakly informative, Wang & Dey (2011)
sp	r	gamma(0.5, 0.5)	Centered near 1 (base distribution)
log_gamma	lambda	normal(0, 1)	Centered at 0 (probit)
aep	theta1	gamma(2, 1)	Mode at 1, symmetric at theta1=theta2
aep	theta2	gamma(2, 1)	Mode at 1, symmetric at theta1=theta2

SP Link Details (Li et al., 2019)

The Symmetric Power link uses a symmetric base distribution F_0, specified via the `base` argument. Supported bases:

- `base = "logit"`: Logistic base (default)
- `base = "probit"`: Normal base
- `base = "cauchit"`: Cauchy base
- `base = "tlink"`: Student-t base (requires df)

Note: Li et al. (2019) define F_0 as a CDF "whose corresponding PDF is symmetric about 0".

`loo.clmstan` *Leave-One-Out Cross-Validation for clmstan objects*

Description

Computes approximate leave-one-out cross-validation (LOO-CV) for a fitted cumulative link model using Pareto smoothed importance sampling (PSIS).

Usage

```
## S3 method for class 'clmstan'
loo(x, ..., r_eff = NULL, cores = getOption("mc.cores", 1), save_psis = FALSE)
```

Arguments

`x`	A `clmstan` object returned by `clm_stan`.
`...`	Additional arguments passed to `loo`.
`r_eff`	A vector of relative effective sample sizes for each observation, or `NULL` (default) to compute them automatically using `relative_eff`. Set to `NA` to skip r_eff computation (faster but diagnostics may be over-optimistic).
`cores`	The number of cores to use for parallel computation. Defaults to `getOption("mc.cores", 1)`.
`save_psis`	If `TRUE`, the PSIS object is saved in the returned object. This is required for some downstream functions like `E_loo()`. Default is `FALSE`.

Details

The function extracts the log-likelihood matrix (`log_lik`) computed in the generated quantities block of the Stan model and passes it to `loo`.

Pareto k diagnostics: Observations with high Pareto k values ($k > 0.7$) indicate potential problems with the LOO approximation for those observations. Use `plot()` on the returned object to visualize the Pareto k values.

Model comparison: Use `loo_compare` to compare multiple models. Models with higher `elpd_loo` are preferred.

Value

An object of class `c("psis_loo", "loo")` containing:

- `estimates`: A matrix with columns `Estimate` and `SE` for `elpd_loo`, `p_loo`, and `looic`.
- `pointwise`: A matrix with pointwise contributions.
- `diagnostics`: A list with Pareto k values and effective sample sizes for each observation.

See Also

`waic.clmstan` for WAIC computation, `loo` for details on the LOO algorithm, `loo_compare` for model comparison.

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
loo_result <- loo(fit)
print(loo_result)
plot(loo_result)

# Compare two models
fit1 <- clm_stan(rating ~ temp, data = wine, link = "logit")
fit2 <- clm_stan(rating ~ temp, data = wine, link = "probit")
loo::loo_compare(loo(fit1), loo(fit2))

## End(Not run)
```

`normal` *Normal Distribution for Prior Specification*

Description

Creates a normal distribution object for use with `prior()`.

Usage

```
normal(mu = 0, sigma = 1)
```

Arguments

mu	Mean of the normal distribution. Default: 0
sigma	Standard deviation of the normal distribution. Must be positive. Default: 1

Value

An object of class "clm_dist" representing a normal distribution.

See Also

prior(), gamma(), student_t(), cauchy()

Examples

```
# Create a normal prior
normal(0, 2.5)

# Use with prior()
prior(normal(0, 2.5), class = "b")
```

---

plot.clmstan *Plot method for clmstan objects*

Description

Produces diagnostic plots using the bayesplot package.

Usage

```
## S3 method for class 'clmstan'
plot(
  x,
  type = c("trace", "dens", "hist", "areas", "intervals", "acf"),
  pars = NULL,
  ...
)
```

Arguments

x	A clmstan object
type	Type of plot: "trace" (default), "dens", "hist", "areas", "intervals", or "acf" (autocorrelation).
pars	Character vector of parameter names to plot. If NULL, plots beta, c_transformed (except first), and beta0.
...	Additional arguments passed to bayesplot functions. For "acf" type, you can use lags to control the number of lags.

Value

A ggplot object

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
plot(fit)                    # trace plots
plot(fit, type = "dens")     # density plots
plot(fit, type = "intervals") # credible intervals
plot(fit, type = "acf")      # autocorrelation plots
plot(fit, pars = "beta")     # only beta parameters

## End(Not run)
```

`posterior_predict.clmstan` *Posterior predictive distribution for clmstan objects*

Description

Draws from the posterior predictive distribution. For each posterior sample, a predicted category is sampled from the categorical distribution with the predicted probabilities.

Usage

```
posterior_predict.clmstan(object, newdata = NULL, ndraws = NULL, ...)
```

Arguments

`object`	A `clmstan` object returned by `clm_stan()`.
`newdata`	Optional data frame for prediction. If `NULL` (default), predictions are made for the original training data.
`ndraws`	Number of posterior draws to use. If `NULL` (default), all available draws are used.
`...`	Additional arguments (currently ignored).

Value

An integer matrix of dimension S x N containing predicted categories (1 to K), where S is the number of posterior draws and N is the number of observations.

See Also

`predict.clmstan()`, `fitted.clmstan()`

predict.clmstan *Predict method for clmstan objects*

Description

Generates predictions from a fitted cumulative link model.

Usage

```
## S3 method for class 'clmstan'
predict(
  object,
  newdata = NULL,
  type = c("class", "probs"),
  summary = TRUE,
  robust = FALSE,
  probs = c(0.025, 0.975),
  ndraws = NULL,
  ...
)
```

Arguments

object	A clmstan object returned by clm_stan().
newdata	Optional data frame for prediction. If NULL (default), predictions are made for the original training data.
type	Type of prediction: • "class": Predicted category (most likely class) • "probs": Predicted probabilities for each category
summary	Logical. If TRUE (default), return summary statistics (mean, SD, quantiles). If FALSE, return raw posterior draws.
robust	Logical. If TRUE, use median instead of mean for point estimates. Default is FALSE.
probs	Numeric vector of probabilities for quantiles. Default is c(0.025, 0.975) for 95% credible intervals.
ndraws	Number of posterior draws to use. If NULL (default), all available draws are used.
...	Additional arguments (currently ignored).

Value

Depending on type and summary:

- type = "class", summary = TRUE: A data frame with columns Estimate (mean/median predicted class), Est.Error (SD), quantile columns, and Class (modal predicted category).

- `type = "class"`, `summary = FALSE`: An S x N integer matrix of predicted categories (1 to K), where S is the number of posterior draws and N is the number of observations.
- `type = "probs"`, `summary = TRUE`: A data frame with columns for each category probability (`P[Y=1]`, `P[Y=2]`, etc.).
- `type = "probs"`, `summary = FALSE`: An S x N x K array of predicted probabilities.

See Also

`fitted.clmstan()` for expected probabilities, `posterior_predict.clmstan()` for posterior predictive samples.

---

print.clm_dist *Print method for clm_dist objects*

Description

Print method for clm_dist objects

Usage

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

Arguments

`x`	A clm_dist object
`...`	Additional arguments (ignored)

Value

Invisibly returns the input `clm_dist` object.

---

print.clm_prior *Print method for clm_prior objects*

Description

Print method for clm_prior objects

Usage

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

Arguments

x	A clm_prior object
...	Additional arguments (ignored)

Value

Invisibly returns the input `clm_prior` object.

---

print.clm_prior_list *Print method for clm_prior_list objects*

Description

Print method for clm_prior_list objects

Usage

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

Arguments

x	A clm_prior_list object
...	Additional arguments (ignored)

Value

Invisibly returns the input `clm_prior_list` object.

---

print.clm_prior_spec *Print method for clm_prior_spec objects*

Description

Print method for clm_prior_spec objects

Usage

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

Arguments

x	A clm_prior_spec object
...	Additional arguments (ignored)

Value

Invisibly returns the input `clm_prior_spec` object.

`print.clmstan` *Print method for clmstan objects*

Description

Print method for clmstan objects

Usage

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

Arguments

`x`	A clmstan object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

`print.summary.clmstan` *Print method for summary.clmstan objects*

Description

Print method for summary.clmstan objects

Usage

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

Arguments

`x`	A summary.clmstan object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

prior *Specify Prior Distributions*

Description

Specify prior distributions for model parameters using distribution functions.

Usage

```
prior(prior, class = "b", coef = "")
```

Arguments

prior	A distribution object created by normal(), gamma(), student_t(), or cauchy().
class	The parameter class. Valid classes are: • "b": Regression coefficients (beta) • "Intercept": Cutpoints/thresholds (flexible) • "c1": First cutpoint (equidistant) • "d": Threshold interval (equidistant) • "cpos": Positive cutpoints (symmetric) • "df": Degrees of freedom (tlink) • "lambda_ao": Lambda parameter (aranda_ordaz) • "lambda_lg": Lambda parameter (log_gamma) • "xi": Xi parameter (gev) • "r": R parameter (sp) • "theta1", "theta2": Theta parameters (aep)
coef	Optional coefficient name (for future extension).

Value

An object of class "clm_prior_spec" representing the prior specification.

See Also

normal(), gamma(), student_t(), cauchy(), clm_prior()

Examples

```
# Specify a normal prior for regression coefficients
prior(normal(0, 2.5), class = "b")

# Specify a gamma prior for degrees of freedom
prior(gamma(2, 0.1), class = "df")

# Combine multiple priors
c(
```

```
  prior(normal(0, 2.5), class = "b"),
  prior(normal(0, 10), class = "Intercept")
)
```

`student_t` *Student-t Distribution for Prior Specification*

Description

Creates a Student-t distribution object for use with `prior()`.

Usage

```
student_t(df = 3, mu = 0, sigma = 1)
```

Arguments

`df`	Degrees of freedom. Must be positive. Default: 3
`mu`	Location parameter. Default: 0
`sigma`	Scale parameter. Must be positive. Default: 1

Value

An object of class `"clm_dist"` representing a Student-t distribution.

See Also

`prior()`, `normal()`, `gamma()`, `cauchy()`

Examples

```
# Create a Student-t prior with heavy tails
student_t(3, 0, 2.5)

# Use with prior()
prior(student_t(3, 0, 2.5), class = "b")
```

summary.clmstan *Summary method for clmstan objects*

Description

Summary method for clmstan objects

Usage

```
## S3 method for class 'clmstan'
summary(object, probs = c(0.025, 0.5, 0.975), digits = 3, ...)
```

Arguments

object	A clmstan object
probs	Quantile probabilities for credible intervals
digits	Number of significant digits for display
...	Additional arguments (ignored)

Value

An object of class "summary.clmstan" containing:

- coefficients: Posterior summary for regression coefficients
- thresholds: Posterior summary for threshold parameters
- beta0: Posterior summary for intercept
- link_params: Posterior summary for link parameters (full model only)
- model_info: Model metadata (formula, link, threshold, K, N, P)

supported_links *Get supported link functions*

Description

Get supported link functions

Usage

```
supported_links(type = c("all", "standard", "flexible"))
```

Arguments

type	Character string specifying which links to return: • "all" (default): All supported link functions • "standard": Standard links without additional parameters • "flexible": Flexible links with additional parameters

Value

A character vector of supported link function names

Examples

```
supported_links()
supported_links("standard")
supported_links("flexible")
```

---

supported_thresholds *Get supported threshold structures*

Description

Get supported threshold structures

Usage

```
supported_thresholds()
```

Value

A character vector of supported threshold structure names

Examples

```
supported_thresholds()
```

`waic.clmstan` *Widely Applicable Information Criterion for clmstan objects*

Description

Computes the Widely Applicable Information Criterion (WAIC) for a fitted cumulative link model.

Usage

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

Arguments

`x`	A `clmstan` object returned by `clm_stan`.
`...`	Additional arguments (currently ignored).

Details

WAIC is an alternative to LOO-CV that is asymptotically equivalent to leave-one-out cross-validation. However, LOO-CV with PSIS is generally preferred because:

- It provides useful diagnostics (Pareto k values)
- It is more robust in finite samples
- It has been shown to be more reliable in practice

For most purposes, `loo.clmstan` is recommended over WAIC.

Value

An object of class `c("waic", "loo")` containing:

- `estimates`: A matrix with columns `Estimate` and `SE` for `elpd_waic`, `p_waic`, and `waic`.
- `pointwise`: A matrix with pointwise contributions.

See Also

`loo.clmstan` for LOO-CV (recommended), `waic` for details on WAIC computation, `loo_compare` for model comparison.

Examples

```
## Not run:
fit <- clm_stan(rating ~ temp, data = wine)
waic_result <- waic(fit)
print(waic_result)

## End(Not run)
```

Index