R topics documented:

mcp-package ................................................. 2
bernoulli .................................................. 3
criterion .................................................. 3
ex_ar ...................................................... 4
ex_binomial ............................................... 5
ex_demo ................................................... 5
ex_fit ..................................................... 6
ex_plateaus ............................................... 6
ex_quadratic .............................................. 7
ex_rel_prior ............................................... 7
ex_trig .................................................... 8
ex_variance ............................................... 8
ex_varying ............................................... 9
hypothesis ............................................... 9
ilogit ..................................................... 11
logit ..................................................... 11
mcp ....................................................... 12
mcpfit-class ............................................. 15
plot.mcpfit ............................................. 16
plot_pars ................................................ 18
print.mcpprior .......................................... 20
sd_to_prec ............................................... 20
summary.mcpfit ......................................... 21

Index ..................................................... 23

mcp-package  Regression with Multiple Change Points

Description

The mcp package provides an interface to fit regression models with multiple change points between generalized linear segments, optionally with per-segment variance and autocorrelation structures.

The main function of mcp is the mcp() function, which uses a formula syntax to specify a wide range of change point models. Based on the supplied data, formulas, and additional information, it writes JAGS code on the fly and use rstan to fit the model, optionally in parallel to speed up sampling. You will need to install JAGS for mcp() to work.

A large number of post-processing methods can be applied. These include

- Summarise fits using summary(), fixef(), and ranef().
- Visualize fits using plot() and individual parameters using plot_pars().
- Test hypotheses using hypothesis() and loo().

Extensive documentation with worked examples is available at the mcp website.
Bernoulli family for mcp

**Description**

Bernoulli family for mcp

**Usage**

bernoulli(link = "logit")

**Arguments**

| link | Link function. |

**criterion**

Compute information criteria for model comparison

**Description**

Takes an mcpfit as input and computes information criteria using loo or WAIC. Compare models using loo_compare and loo_model_weights. more in loo.

**Usage**

criterion(fit, criterion = "loo", ...)

## S3 method for class 'mcpfit'

loo(x, ...)

## S3 method for class 'mcpfit'

waic(x, ...)

**Arguments**

| fit  | An mcpfit object. |

| criterion | One of "loo" (calls loo) or "waic" (calls waic). |

| ... | Currently ignored |

| x | An mcpfit object. |

**Value**

a loo or psis_loo object.
Methods (by generic)

- `loo`: Computes loo on mcpfit objects
- `waic`: Computes WAIC on mcpfit objects

Author(s)

Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

See Also

criterion
criterion

Examples

```r
# Define two models and sample them
# options(mc.cores = 3) # Speed up sampling
model1 = list(y ~ 1 + x, ~ 1)
model2 = list(y ~ 1 + x) # Without a change point
fit1 = mcp(model1, ex_plateaus)
fit2 = mcp(model2, ex_plateaus)

# Compute L00 for each and compare (works for waic(fit) too)
fit1$loo = loo(fit1)
fit2$loo = loo(fit2)
loo::loo_compare(fit1$loo, fit2$loo)
```

Description

See how this data was simulated here using `fit$simulate()`, including which parameters were used. See an analysis here.

Usage

ex_ar

Format

A data frame with 300 rows and 2 variables:

- `time` The x-axis.
- `price` The y-axis.
**ex_binomial**

Two change points between three binomial segments

**Description**
See how this data was simulated here using `fit$simulate()`, including which parameters were used. See an analysis here.

**Usage**
ex_binomial

**Format**
A data frame with 100 rows and 3 variables:

- **x** The x-axis.
- **y** The y-axis.
- **N** The number of trials for `y`.

---

**ex_demo**

Two change points between three linear segments

**Description**
See how this data was simulated here using `fit$simulate()`, including which parameters were used. See an analysis here.

**Usage**
ex_demo

**Format**
A data frame with 100 rows and 2 variables:

- **time** The x-axis.
- **response** The y-axis.

**Details**
The `ex_fit` object is fitted to this dataset.
Example fit of the `ex_demo` dataset

Usage

```r
ex_fit
```

Format

An `mcpfit` object.

---

A change point between two plateaus

Description

See how this data was simulated [here](#) using `fit$simulate()`, including which parameters were used. See an analysis [here](#).

Usage

```r
ex_plateaus
```

Format

A data frame with 100 rows and 2 variables:

- `x` The x-axis.
- `y` The y-axis.
ex_quadratic

A change point from plateau to quadratic

Description
See how this data was simulated here using fit$simulate(), including which parameters were used. See an analysis here.

Usage
ex_quadratic

Format
A data frame with 81 rows and 2 variables:

x The x-axis.
y The y-axis.

ex_rel_prior

Two change points between three linear segments

Description
See how this data was simulated here using fit$simulate(), including which parameters were used. See an analysis here.

Usage
ex_rel_prior

Format
A data frame with 100 rows and 2 variables:

x The x-axis.
y The y-axis.
### ex_trig

*A change point between two trigonometric segments*

#### Description

See how this data was simulated [here](#) using `fit$simulate()`, including which parameters were used. See an analysis [here](#).

#### Usage

`ex_trig`

#### Format

A data frame with 234 rows and 2 variables:

- **x**: The x-axis.
- **y**: The y-axis.

### ex_variance

*Two change points between three heteroskedastic segments*

#### Description

See how this data was simulated [here](#) using `fit$simulate()`, including which parameters were used. See an analysis [here](#).

#### Usage

`ex_variance`

#### Format

A data frame with 100 rows and 2 variables:

- **x**: The x-axis.
- **y**: The y-axis.
Description

See how this data was simulated here using `fit$simulate()`, including which parameters were used. See an analysis here.

Usage

```r
ex_varying
```

Format

A data frame with 150 rows and 4 variables:

- `x` The x-axis.
- `y` The y-axis.
- `id` The participant id (character).
- `id_numeric` The participant id (numeric).

Description

Test hypotheses on `mcp` objects.

Usage

```r
hypothesis(fit, hypotheses, width = 0.95, digits = 3)
```
Arguments

fit  
An `mcpfit` object.

hypotheses  
String representation of a logical test involving model parameters. Takes R code that evaluates to TRUE or FALSE in a vectorized way. Directional hypotheses are specified using `<`, `>`, `<=`, or `>=`. `hypothesis` returns the posterior probability and odds in favor of the stated hypothesis. The odds can be interpreted as a Bayes Factor. For example:

- "cp_1 > 30": the first change point is above 30.
- "int_1 > int_2": the intercept is greater in segment 1 than 2.
- "x_2 - x_1 <= 3": the difference between slope 1 and 2 is less than or equal to 3.
- "int_1 > -2 & int_1 < 2": int_1 is between -2 and 2 (an interval hypothesis). This can be useful as a Region Of Practical Equivalence test (ROPE).
- "cp_1^2 < 30 | (log(x_1) + log(x_2)) > 5": be creative.
- `\`cp_1_id[1]\` > `\`cp_1_id[2]\`: `id1` is greater than `id2`, as estimated through the varying-by-"id" change point in segment 1. Note that `\` required for varying effects.

Hypotheses can also test equality using the equal sign (=). This runs a Savage-Dickey test, i.e., the proportion by which the probability density has increased from the prior to the posterior at a given value. Therefore, it requires `mcp(sample = "both")`. There are two requirements: First, there can only be one equal sign, so don’t use and (`&`) or or (`|`). Second, the point to test has to be on the right, and the variables on the left.

- "cp_1 = 30": is the first change point at 30? Or to be more precise: by what factor has the credence in `cp_1 = 30` risen/fallen when conditioning on the data, relative to the prior credence?
- "int_1 + int_2 = 0": Is the sum of two intercepts zero?
- ``'cp_1_id[John]' > 'cp_1_id[Erin]' = 2``": is the varying change point for John (which is relative to `cp_1") double that of Erin?

width  
Float. The width of the highest posterior density interval (between 0 and 1).

digits  
a non-null value for `digits` specifies the minimum number of significant digits to be printed in values. The default, NULL, uses `getOption("digits")`. (For the interpretation for complex numbers see `signif"). Non-integer values will be rounded down, and only values greater than or equal to 1 and no greater than 22 are accepted.

Value

A data.frame with a row per hypothesis and the following columns:

- hypothesis is the hypothesis; often re-arranged to test against zero.
- mean is the posterior mean of the left-hand side of the hypothesis.
- lower is the lower bound of the (two-sided) highest-density interval of width `width`.
- upper is the upper bound of ditto.
• p Posterior probability. For "=" (Savage-Dickey), it is the BF converted to p. For directional hypotheses, it is the proportion of samples that returns TRUE.

• BF Bayes Factor in favor of the hypothesis. For "=" it is the Savage-Dickey density ratio. For directional hypotheses, it is p converted to odds.

Author(s)
Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

ilogit

Inverse logit function

Description
Inverse logit function

Usage
ilogit(eta)

Arguments
eta A vector of logits

Value
A vector with same length as eta

logit

Logit function

Description
Logit function

Usage
logit(mu)

Arguments
mu A vector of probabilities (0-1)

Value
A vector with same length as mu
mcp

Fit Multiple Linear Segments And Their Change Points

Description

Given a model (a list of segment formulas), mcp infers the posterior distributions of the parameters of each segment as well as the change points between segments. See more details and worked examples on the mcp website. All segments must regress on the same x-variable. Change points are forced to be ordered using truncation of the priors. You can run `fit = mcp(model, sample=FALSE)` to avoid sampling and the need for data if you just want to get the priors (`fit$prior`), the JAGS code `fit$jags_code`, or the R function to simulate data (`fit$simulate`).

Usage

```r
mcp(
  model, data = NULL,
  prior = list(),
  family = gaussian(),
  par_x = NULL,
  sample = "post",
  cores = 1,
  chains = 3,
  iter = 3000,
  adapt = 1500,
  inits = NULL,
  jags_code = NULL
)
```

Arguments

- **model**: A list of formulas - one for each segment. The first formula has the format `response ~ predictors` while the following formulas have the format `response ~ changepoint ~ predictors`. The response and change points can be omitted (`changepoint ~ predictors` assumes same response. `~ predictors` assumes an intercept-only change point). The following can be modeled:
  - **Regular formulas**: e.g., `~ 1 + x`). Read more.
  - **Extended formulas**: e.g., `~ I(x^2) + exp(x) + sin(x)`. Read more.
  - **Variance**: e.g., `~sigma(1)` for a simple variance change or `~sigma(rel(1) + I(x^2))` for more advanced variance structures. Read more
  - **Autoregression**: e.g., `~ar(1)` for a simple onset/change in AR(1) or `ar(2, 0 + x)` for an AR(2) increasing by x. Read more
- **data**: Data.frame or tibble in long format.
- **prior**: Named list. Names are parameter names (`cp_i, int_i, xvar_i, 'sigma'`) and the values are either
• A JAGS distribution (e.g., int_1 = "dnorm(0,1) T(0,)") indicating a conventional prior distribution. Uninformative priors based on data properties are used where priors are not specified. This ensures good parameter estimates, but it is questionable for hypothesis testing. mcp uses SD (not precision) for dnorm, dt, dlogis, etc. See details. Change points are forced to be ordered through the priors using truncation, except for uniform priors where the lower bound should be greater than the previous change point, dunif(cp_1, MAXX).

• A numerical value (e.g., int_1 = -2.1) indicating a fixed value.

• A model parameter name (e.g., int_2 = "int_1"), indicating that this parameter is shared - typically between segments. If two varying effects are shared this way, they will need to have the same grouping variable.

• A scaled Dirichlet prior is supported for change points if they are all set to cp_i = "dirichlet(N) where N is the alpha for this change point and N = 1 is most often used. This prior is less informative about the location of the change points than the default uniform prior, but it samples less efficiently, so you will often need to set iter higher. It is recommended for hypothesis testing and for the estimation of more than 5 change points. Read more.

family

• One of gaussian(), binomial(), bernoulli(), or poission(). Only default link functions are currently supported.

par_x

• String (default: NULL). Only relevant if no segments contains slope (no hint at what x is). Set this, e.g., par_x = "time".

sample

• "post" (default): Sample the posterior.
• "prior": Sample only the prior. Plots, summaries, etc. will use the prior. This is useful for prior predictive checks.
• "both": Sample both prior and posterior. Plots, summaries, etc. will default to using the posterior. The prior only has effect when doing Savage-Dickey density ratios in hypothesis.
• "none" or FALSE: Do not sample. Returns an mcpfit object without sample. This is useful if you only want to check prior strings (fit$prior), the JAGS model (fit$jags_code), etc.

cores

• Positive integer or "all". Number of cores.
• 1: serial sampling
• >1: parallel sampling on this number of cores. Ideally set chains to the same value. Note: cores > 1 takes a few extra seconds the first time it’s called but subsequent calls will start sampling immediately.
• "all": use all cores but one and sets chains to the same value. This is a convenient way to maximally use your computer’s power.

chains

• Positive integer. Number of chains to run.

iter

• Positive integer. Number of post-warmup samples to draw.

adapt

• Positive integer. Also sometimes called "burnin", this is the number of samples used to reach convergence. Set lower for greater speed. Set higher if the chains haven’t converged yet or look at tips, tricks, and debugging.
inits            A list if initial values for the parameters. This can be useful if a model fails to converge. Read more in `jags.model`. Defaults to NULL, i.e., no inits.

jags_code       Pass JAGS code to `mcp` to use directly. This is useful if you want to tweak the code in `fit$jags_code` and run it within the `mcp` framework.

Details

Notes on priors:

- Order restriction is automatically applied to `cp_*` parameters using truncation (e.g., `T(cp_1,)`) so that they are in the correct order on the x-axis UNLESS you do it yourself. The one exception is for dunif distributions where you have to do it as above.

- In addition to the model parameters, MINX (minimum x-value), MAXX (maximum x-value), SDX (etc...), MINY, MAXY, and SDY are also available when you set priors. They are used to set uninformative default priors.

- Use SD when you specify priors for dt, dlogis, etc. JAGS uses precision but `mcp` converts to precision under the hood via the `sd_to_prec()` function. So you will see SDs in `fit$prior` but precision ($1/SD^2$) in `fit$jags_code`

Value

An `mcpfit` object.

Author(s)

Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

See Also

generate_segment_table

Examples

```r
# Define the segments using formulas. A change point is estimated between each formula.
model = list(
  response ~ 1,  # Plateau in the first segment (int_1)
  ~ 0 + time,   # Joined slope (time_2) at cp_1
  ~ 1 + time    # Disjoined slope (int_3, time_3) at cp_2
)

# Fit it. The 'ex_demo' dataset is included in mcp. Sample the prior too.
# options(mc.cores = 3)  # Uncomment to speed up sampling
ex_fit = mcp(model, data = ex_demo, sample = "both")

# See parameter estimates
summary(ex_fit)

# Visual inspection of the results
plot(ex_fit)
```
plot_pars(ex_fit)

# Test a hypothesis
hypothesis(ex_fit, "cp_1 > 10")

# Compare to a one-intercept-only model (no change points) with default prior
model_null = list(response ~ 1)
fit_null = mcp(model_null, data = ex_demo, par_x = "time")  # fit another model here
ex_fit$loo = loo(ex_fit)
fit_null$loo = loo(fit_null)
loo::loo_compare(ex_fit$loo, fit_null$loo)

# Inspect the prior. Useful for prior predictive checks.
summary(ex_fit, prior = TRUE)
plot(ex_fit, prior = TRUE)

# Show all priors. Default priors are added where you don't provide any
print(ex_fit$prior)

# Set priors and re-run
prior = list(
  int_1 = 15,
  time_2 = "dt(0, 2, 1) T(0, )",  # t-dist slope. Truncated to positive.
  cp_2 = "dunif(cp_1, 80)",  # change point to segment 2 > cp_1 and < 80.
  int_3 = "int_1"  # Shared intercept between segment 1 and 3
)

fit3 = mcp(model, data = ex_demo, prior = prior)

# Show the JAGS model
cat(ex_fit$jags_code)

---

mcpfit-class

Class mcpfit of models fitted with the mcp package

Description

Models fitted with the mcp function are represented as an mcpfit object which contains the user input (model, data, family), derived model characteristics (prior, parameter names, and jags code), and the fit (prior and/or posterior mcmc samples).

Details

See methods(class = "mcpfit") for an overview of available methods.

User-provided information (see mcp for more details):
Slots

- **model**  A list of formulas, making up the model. Provided by user. See `mcp` for more details.
- **data**   A data frame. Provided by user. See `mcp` for more details.
- **family** An `mcpfamily` object. Provided by user. See `mcp` for more details.
- **prior**  A named list. Provided by user. See `mcp` for more details.
- **mcmc_post** An `mcmc.list` object with posterior samples.
- **mcmc_prior** An `mcmc.list` object with prior samples.
- **mcmc_loglik** An `mcmc.list` object with samples of log-likelihood.
- **pars**   A list of character vectors of model parameter names.
- **jags_code** A string with jags code. Use `cat(fit$jags_code)` to show it.
- **simulate** A method to simulate and predict data.
- **.other** Information that is used internally by `mcp`.

---

**plot.mcpfit**  
*Plot full fits*

---

**Description**

Plot prior or posterior model draws on top of data. Use `plot_pars` to plot individual parameter estimates.

**Usage**

```r
## S3 method for class 'mcpfit'
plot(
  x,
  facet_by = NULL,
  lines = 25,
  geom_data = "point",
  cp_dens = TRUE,
  q_fit = FALSE,
  q_predict = FALSE,
  rate = TRUE,
  prior = FALSE,
  which_y = "ct",
  ...
)
```

```r
```
Arguments

- **x**: An `mcpfit` object
- **facet_by**: String. Name of a varying group.
- **lines**: Positive integer or `FALSE`. Number of lines (posterior draws). `FALSE` or `lines = 0` plots no lines. Note that lines always plot fitted values - not predicted. For prediction intervals, see the `q_predict` argument.
- **geom_data**: String. One of "point" (default), "line" (good for time-series), or `FALSE` (don't plot).
- **cp_dens**: `TRUE/FALSE`. Plot posterior densities of the change point(s)? Currently does not respect `facet_by`. This will be added in the future.
- **q_fit**: Whether to plot quantiles of the posterior (fitted value).
  - `TRUE`: Add 2.5% and 97.5% quantiles. Corresponds to `q_fit = c(0.025, 0.975)``.
  - `FALSE` (default): No quantiles
  - A vector of quantiles. For example, `quantiles = 0.5` plots the median and `quantiles = c(0.2, 0.8)` plots the 20% and 80% quantiles.
- **q_predict**: Same as `q_fit`, but for the prediction interval.
- **rate**: Boolean. For binomial models, plot on raw data (`rate = FALSE`) or response divided by number of trials (`rate = TRUE`). If `FALSE`, linear interpolation on trial number is used to infer trials at a particular `x`.
- **prior**: `TRUE/FALSE`. Plot using prior samples? Useful for `mcp(..., sample = "both")`
- **which_y**: What to plot on the y-axis. One of
  - "ct": The central tendency which is often the mean after applying the link function (default).
  - "sigma": The variance
  - "ar1", "ar2", etc. depending on which order of the autoregressive effects you want to plot.

... Currently ignored.

Details

`plot()` uses `fit$simulate()` on posterior samples. These represent the (joint) posterior distribution.

Value

A `ggplot2` object.

Author(s)

Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>
**Examples**

```r
# Typical usage. ex_fit is an mcpfit object.
plot(ex_fit)
plot(ex_fit, prior = TRUE)  # The prior

plot(ex_fit, lines = 0, q_fit = TRUE)  # 95% HDI without lines
plot(ex_fit, q_predict = c(0.1, 0.9))  # 80% prediction interval
plot(ex_fit, which_y = "sigma", lines = 100)  # The variance parameter on y

# Show a panel for each varying effect
# plot(fit, facet_by = "my_column")

# Customize plots using regular ggplot2
library(ggplot2)
plot(ex_fit) + theme_bw(15) + ggtitle("Great plot!")
```

---

**plot_pars**  
*Plot individual parameters*

**Description**

Plot many types of plots of parameter estimates. See examples for typical use cases.

**Usage**

```r
plot_pars(
  fit,  
  pars = "population",  
  regex_pars = character(0),  
  type = "combo",  
  ncol = 1,  
  prior = FALSE
)
```

**Arguments**

- **fit**: An `mcpfit` object.
- **pars**: Character vector. One of:
  - Vector of parameter names.
  - "population" (default): plots all population parameters.
  - "varying": plots all varying effects. To plot a particular varying effect, use `regex_pars = "^name"`.
- **regex_pars**: Vector of regular expressions. This will typically just be the beginning of the parameter name(s), i.e., "^cp_" plots all change points, "^my_varying" plots all levels of a particular varying effect, and "^cp_\^my_varying" plots both.
**plot_pars**

String or vector of strings. Calls bayesplot::mcmc_>>type<<(). Common calls are "combo", "trace", and "dens_overlay". Current options include 'acf', 'acf_bar', 'areas', 'areas_ridges', 'combo', 'dens', 'dens_chains', 'dens_overlay', 'hist', 'intervals', 'rank_hist', 'rank_overlay', 'trace', 'trace_highlight', and 'violin'.

Number of columns in plot. This is useful when you have many parameters and only one plot type.

TRUE/FALSE. Plot using prior samples? Useful for mcp(..., sample = "both")

For other **type**, it calls bayesplot::mcmc_type(). Use these directly on fit$mcmc_post or fit$mcmc_prior if you want finer control of plotting, e.g., bayesplot::mcmc_dens(fit$mcmc_post). There are also a number of useful plots in the coda package, i.e., coda::gelman.plot(fit$mcmc_post) and coda::crosscorr.plot(fit$mcmc_post)

In any case, if you see a few erratic lines or parameter estimates, this is a sign that you may want to increase argument 'adapt' and 'iter' in mcp.

A ggplot2 object.

Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

# Typical usage. ex_fit is an mcpfit object.
plot_pars(ex_fit)

# More options
plot_pars(ex_fit, regex_pars = "^cp_") # Plot only change points
plot_pars(ex_fit, pars = c("int_3", "time_3")) # Plot these parameters
plot_pars(ex_fit, type = c("trace", "violin")) # Combine plots

# Not run:
# Some plots only take pairs. hex is good to assess identifiability
plot_pars(ex_fit, type = "hex", pars = c("cp_1", "time_2"))

# Visualize the priors:
plot_pars(ex_fit, prior = TRUE)

# Useful for varying effects:
# plot_pars(my_fit, pars = "varying", ncol = 3) # plot all varying effects
# plot_pars(my_fit, regex_pars = "my_varying", ncol = 3) # plot all levels of a particular varying

# Customize multi-column ggplots using "+" instead of "*" (patchwork)
library(ggplot2)
plot_pars(ex_fit, type = c("trace", "dens_overlay")] * theme_bw(10)

# End(Not run)
### Description

The `mcpprior` is just a list, but it can be displayed in a more condensed way using `cbind`.

### Usage

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

### Arguments

- **x**: An `mcpfit` object.
- **...**: Currently ignored

---

### Description

JAGS uses precision rather than SD. This function converts `dnorm(4.2,1.3)` into `dnorm(4.2,1/1.3^2)`. It allows users to specify priors using SD and then it’s transformed for the JAGS code. It works for the following distributions: `dnorm|dt|dcauchy|ddexp|dlogis|dlnorm`. In all of these, tau/sd is the second parameter.

### Usage

```
sd_to_prec(prior_str)
```

### Arguments

- **prior_str**: String. A JAGS prior. Can be truncated, e.g. `dt(3, 2, 1)` T(my_var, )

### Value

A string

### Author(s)

Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>
### summary.mcpfit

**Summarise mcpfit objects**

**Description**

Summarise parameter estimates and model diagnostics.

**Usage**

```r
## S3 method for class 'mcpfit'
summary(object, width = 0.95, digits = 2, prior = FALSE, ...)
```

```r
fixef(object, width = 0.95, prior = FALSE, ...)
```

```r
ranef(object, width = 0.95, prior = FALSE, ...)
```

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

**Arguments**

- `object`: An `mcpfit` object.
- `width`: Float. The width of the highest posterior density interval (between 0 and 1).
- `digits`: a non-null value for `digits` specifies the minimum number of significant digits to be printed in values. The default, NULL, uses `getOption("digits")`. (For the interpretation for complex numbers see `signif`.) Non-integer values will be rounded down, and only values greater than or equal to 1 and no greater than 22 are accepted.
- `prior`: TRUE/FALSE. Summarise prior instead of posterior?
- `...`: Currently ignored
- `x`: An `mcpfit` object.

**Value**

A data frame with parameter estimates and MCMC diagnostics. OBS: The change point distributions are often not unimodal and symmetric so the intervals can be deceiving. Plot them using `plot_pars(fit)`.

- `mean` is the posterior mean
- `lower` is the lower quantile of the highest-density interval (HDI) given in `width`.
- `upper` is the upper quantile.
- `Rhat` is the Gelman-Rubin convergence diagnostic which is often taken to be acceptable if < 1.1. It is computed using `gelman.diag`. 
• n.eff is the effective sample size computed using `effectiveSize`. Low effective sample sizes are also obvious as poor mixing in trace plots (see `plot_pars(fit)`). Read how to deal with such problems here.
• ts_err is the time-series error, taking autoregressive correlation into account. It is computed using `spectrum0.ar`.

For simulated data, the summary contains two additional columns so that it is easy to inspect whether the model can recover the parameters. Run simulation and summary multiple times to get a sense of the robustness.

• sim is the value used to generate the data.
• match is "OK" if sim is contained in the HDI interval (lower to upper).

Functions
• fixef: Get population-level ("fixed") effects of an `mcpfit` object.
• ranef: Get varying ("random") effects of an `mcpfit` object.
• `print.mcpfit`: Print the posterior summary of an `mcpfit` object.

Author(s)
Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

Examples

```r
# Typical usage
summary(ex_fit)
summary(ex_fit, width = 0.8, digits = 4) # Set HDI width

# Get the results as a data frame
results = summary(ex_fit)

# Varying (random) effects
# ranef(my_fit)

# Summarise prior
summary(ex_fit, prior = TRUE)
```
Index

*Topic datasets
  ex_ar, 4
  ex_binomial, 5
  ex_demo, 5
  ex_fit, 6
  ex_plateaus, 6
  ex_quadratic, 7
  ex_rel_prior, 7
  ex_trig, 8
  ex_variance, 8
  ex_varying, 9

bernoulli, 3

criterion, 3, 4

effectiveSize, 22
ex_ar, 4
ex_binomial, 5
ex_demo, 5
ex_fit, 6
ex_plateaus, 6
ex_quadratic, 7
ex_rel_prior, 7
ex_trig, 8
ex_variance, 8
ex_varying, 9

fixed.effects (summary.mcpfit), 21
fixef (summary.mcpfit), 21

gelman.diag, 21
get_segment_table, 14

hypothesis, 9, 13

ilogit, 11

jags.model, 14

logit, 11

LOO (criterion), 3
loo, 3
loo (criterion), 3
loo_compare, 3
loo_model_weights, 3

mcmc.list, 16
mcp, 12, 15, 16, 19
mcp-package, 2
mcpfit, 3, 6, 10, 14, 17, 18, 20–22
mcpfit (mcpfit-class), 15
mcpfit-class, 15

plot (plot.mcpfit), 16
plot.mcpfit, 16
plot_pars, 18
print (summary.mcpfit), 21
print.mcpprior, 20

random.effects (summary.mcpfit), 21
ranef (summary.mcpfit), 21

sd_to_prec, 20
spectrum0.ar, 22
summary (summary.mcpfit), 21
summary.mcpfit, 21

WAIC (criterion), 3
waic, 3
waic (criterion), 3