Package 'pintervals'

May 6, 2025

Type Package

Title Model Agnostic Prediction Intervals

Version 0.7.7

Description Provides tools for estimating model-agnostic prediction intervals using conformal prediction, bootstrapping, and parametric prediction intervals. The package is designed for ease of use, offering intuitive functions for both binned and full conformal prediction methods, as well as parametric interval estimation with diagnostic checks. Currently only working for continuous predictions. For details on the conformal and bin-conditional conformal prediction methods, see Randahl, Williams, and Hegre (2024) <DOI:10.48550/arXiv.2410.14507>.

License MIT + file LICENSE

Encoding UTF-8

Imports dplyr, foreach, tibble

RoxygenNote 7.3.2

NeedsCompilation no

Author David Randahl [aut, cre],
Jonathan P. Williams [ctb]

Maintainer David Randahl <david.randahl@pcr.uu.se>

Repository CRAN

Date/Publication 2025-05-06 16:10:02 UTC

Contents

abs_error 2
bindividual_alpha 2
bin_chopper 3
bootstrap_inner 3
contiguize_intervals 4
flatten_cp_bin_intervals 4
grid_finder 5
grid_inner 6
minq_to_alpha 6

pinterval_bcbootstrap 7
pinterval_bccp 8
pinterval_bootstrap 11
pinterval_conformal 12
pinterval_parametric 14
weights_calculator 16

Index **17**

`abs_error` *Absolute Error*

Description

Absolute Error

Usage

```
abs_error(pred, truth)
```

Arguments

`pred`	a numeric vector of predicted values
`truth`	a numeric vector of true values

Value

a numeric vector of absolute errors

`bindividual_alpha` *Bin-individual alpha function for conformal prediction*

Description

Bin-individual alpha function for conformal prediction

Usage

```
bindividual_alpha(minqs, alpha)
```

Arguments

`minqs`	Minimum quantiles
`alpha`	alpha level

`bin_chopper` *Bin chopper function for binned bootstrapping*

Description

Bin chopper function for binned bootstrapping

Usage

```
bin_chopper(x, nbins, return_breaks = FALSE)
```

Arguments

`x`	vector of values to be binned
`nbins`	number of bins
`return_breaks`	logical indicating whether to return the bin breaks

`bootstrap_inner` *Bootstrap function for bootstrapping the prediction intervals*

Description

Bootstrap function for bootstrapping the prediction intervals

Usage

```
bootstrap_inner(pred, error, nboot, alpha, lower_bound, upper_bound)
```

Arguments

`pred`	predicted value
`error`	vector of errors
`nboot`	number of bootstrap samples
`alpha`	confidence level
`lower_bound`	lower bound of the prediction interval
`upper_bound`	upper bound of the prediction interval

Value

a numeric vector with the predicted value and the lower and upper bounds of the prediction interval

`contiguize_intervals` *Contiguize non-contiguous intervals*

Description

Contiguize non-contiguous intervals

Usage

```
contiguize_intervals(
  pot_lower_bounds,
  pot_upper_bounds,
  empirical_lower_bounds,
  empirical_upper_bounds,
  return_all = FALSE
)
```

Arguments

`pot_lower_bounds`	Potential non-contiguous lower bounds
`pot_upper_bounds`	Potential non-contiguous upper bounds
`empirical_lower_bounds`	Observed lower bounds
`empirical_upper_bounds`	Observed upper bounds
`return_all`	Return all intervals or just contiguous intervals

`flatten_cp_bin_intervals` *Flatten binned conformal prediction intervals to contiguous intervals*

Description

Flatten binned conformal prediction intervals to contiguous intervals

Usage

```
flatten_cp_bin_intervals(lst, contiguize = FALSE)
```

Arguments

`lst`	list of binned conformal prediction intervals
`contiguize`	logical indicating whether to contiguize the intervals

`grid_finder`	*Grid search for lower and upper bounds of continuous conformal prediction intervals*

Description

Grid search for lower and upper bounds of continuous conformal prediction intervals

Usage

```
grid_finder(
  y_min,
  y_max,
  ncs,
  ncs_function,
  y_hat,
  alpha,
  min_step = NULL,
  grid_size = NULL,
  return_min_q = FALSE,
  weighted_cp = FALSE,
  calib = NULL
)
```

Arguments

`y_min`	minimum value to search
`y_max`	maximum value to search
`ncs`	vector of non-conformity scores
`ncs_function`	a function that takes a vector of predicted values and a vector of true values and returns a vector of non-conformity scores
`y_hat`	vector of predicted values
`alpha`	confidence level
`min_step`	The minimum step size for the grid search
`grid_size`	Alternative to min_step, the number of points to use in the grid search between the lower and upper bound
`return_min_q`	logical. If TRUE, the function will return the minimum quantile of the nonconformity scores for each predicted value
`weighted_cp`	logical. If TRUE, the function will use the weighted conformal prediction method. Default is FALSE
`calib`	a tibble with the predicted values and the true values of the calibration partition. Used when weighted_cp is TRUE. Default is NULL

Value

a tibble with the predicted values and the lower and upper bounds of the prediction intervals

grid_inner *Inner function for grid search*

Description

Inner function for grid search

Usage

```
grid_inner(
  hyp_ncs,
  y_hat,
  ncs,
  pos_vals,
  alpha,
  return_min_q = FALSE,
  weights = NULL
)
```

Arguments

hyp_ncs	vector of hypothetical non-conformity scores
y_hat	predicted value
ncs	vector of non-conformity scores
pos_vals	vector of possible values for the lower and upper bounds of the prediction interval
alpha	confidence level
return_min_q	logical. If TRUE, the function will return the minimum quantile of the nonconformity scores for each predicted value
weights	vector of weights for the weighted conformal prediction method

Value

a numeric vector with the predicted value and the lower and upper bounds of the prediction interval

minq_to_alpha *Helper for minimum quantile to alpha function*

Description

Helper for minimum quantile to alpha function

Usage

```
minq_to_alpha(minq, alpha)
```

Arguments

`minq`	minimum quantile
`alpha`	alpha level

`pinterval_bcbootstrap` *Bin-conditional bootstrap prediction intervals*

Description

This function computes bootstrapped prediction intervals with a confidence level of 1-alpha for a vector of (continuous) predicted values using bin-conditional bootstrapped prediction errors. The prediction errors to bootstrap from are computed using either a calibration set with predicted and true values or a set of pre-computed prediction errors from a calibration dataset or other data which the model was not trained on (e.g. OOB errors from a model using bagging). The function returns a tibble containing the predicted values along with the lower and upper bounds of the prediction intervals.

Currently not working as intended. May be removed in future versions.

Usage

```
pinterval_bcbootstrap(
  pred,
  calib,
  calib_truth = NULL,
  calib_bins = NULL,
  breaks = NULL,
  nbins = NULL,
  calib_bin_type = c("prediction", "truth"),
  error_type = c("raw", "absolute"),
  alpha = 0.1,
  n_bootstraps = 1000,
  lower_bound = NULL,
  upper_bound = NULL,
  right = TRUE
)
```

Arguments

`pred`	Vector of predicted values
`calib`	A numeric vector of predicted values in the calibration partition or a 2 column tibble or matrix with the first column being the predicted values and the second column being the truth values

`calib_truth`	A numeric vector of true values in the calibration partition. Only required if calib is a numeric vector
`calib_bins`	A vector of bin identifiers for the calibration set
`breaks`	A vector of break points for the bins to manually define the bins. If NULL, lower and upper bounds of the bins are calculated as the minimum and maximum values of each bin in the calibration set. Must be provided if calib_bins or nbins are not provided, either as a vector or as the last column of a calib tibble.
`nbins`	Automatically chop the calibration set into nbins based on the true values with approximately equal number of observations in each bin. Must be provided if calib_bins or breaks are not provided.
`calib_bin_type`	A string specicying whether the bins are based on the predicted values ('prediction') or the true values ('truth'). Default is 'prediction'. Ignored if calib_bins is provided.
`error_type`	The type of error to use for the prediction intervals. Can be 'raw' or 'absolute'. If 'raw', bootstrapping will be done on the raw prediction errors. If 'absolute', bootstrapping will be done on the absolute prediction errors with random signs. Default is 'raw'
`alpha`	The confidence level for the prediction intervals. Must be a single numeric value between 0 and 1
`n_bootstraps`	The number of bootstraps to perform. Default is 1000
`lower_bound`	Optional minimum value for the prediction intervals. If not provided, the minimum (true) value of the calibration partition will be used
`upper_bound`	Optional maximum value for the prediction intervals. If not provided, the maximum (true) value of the calibration partition will be used
`right`	Parameter passed to cut function to determine which side of the bin interval is closed. Default is TRUE

`pinterval_bccp` *Bin-conditional conformal prediction intervals for continuous predictions*

Description

This function calculates bin-conditional conformal prediction intervals with a confidence level of 1-alpha for a vector of (continuous) predicted values using inductive conformal prediction on a bin-by-bin basis. The intervals are computed using either a calibration set with predicted and true values or a set of pre-computed non-conformity scores from the calibration set. In addition the function requires either a set of breaks or a vector of bin identifiers for the calibrations set, either as a standalone vector or as the third column of the calibration dataset if the calibration data is provided as a tibble. The function returns a tibble containing the predicted values along with the lower and upper bounds of the prediction intervals. Bin-conditional conformal prediction intervals are useful when the prediction error is not constant across the range of predicted values and ensures that the coverage is (approximately) correct for each bin under the assumption that the non-conformity scores are exchangeable within each bin.

Usage

```
pinterval_bccp(
  pred,
  calib = NULL,
  calib_truth = NULL,
  calib_bins = NULL,
  breaks = NULL,
  nbins = NULL,
  alpha = 0.1,
  ncs_function = "absolute_error",
  ncs = NULL,
  min_step = 0.01,
  grid_size = NULL,
  right = TRUE,
  weighted_cp = FALSE,
  contiguize = FALSE
)
```

Arguments

`pred`	Vector of predicted values
`calib`	A numeric vector of predicted values in the calibration partition or a 2 or 3 column tibble or matrix with the first column being the predicted values and the second column being the truth values and (optionally) the third column being the bin values if bins are not provided as a standalone vector or if breaks are not provided
`calib_truth`	A numeric vector of true values in the calibration partition
`calib_bins`	A vector of bin identifiers for the calibration set
`breaks`	A vector of break points for the bins to manually define the bins. If NULL, lower and upper bounds of the bins are calculated as the minimum and maximum values of each bin in the calibration set. Must be provided if calib_bins or nbins are not provided, either as a vector or as the last column of a calib tibble.
`nbins`	Automatically chop the calibration set into nbins based on the true values with approximately equal number of observations in each bin. Must be provided if calib_bins or breaks are not provided.
`alpha`	The confidence level for the prediction intervals. Must be a single numeric value between 0 and 1
`ncs_function`	A function or a character string matching a function that takes two arguments, a vector of predicted values and a vector of true values, in that order. The function should return a numeric vector of nonconformity scores. Default is 'absolute_error' which returns the absolute difference between the predicted and true values.
`ncs`	An optional numeric vector of pre-computed nonconformity scores from a calibration partition. If provided, calib will be ignored. If provided, bins must be provided in calib_bins and breaks as well.

`min_step`	The minimum step size for the grid search. Default is 0.01. Useful to change if predictions are made on a discrete grid or if the resolution of the interval is too coarse or too fine.
`grid_size`	Alternative to min_step, the number of points to use in the grid search between the lower and upper bound. If provided, min_step will be ignored.
`right`	Logical, if TRUE the bins are right-closed (a,b] and if FALSE the bins are left-closed '[a,b)'. Only used if breaks or nbins are provided.
`weighted_cp`	Logical, if TRUE the prediction intervals are created by bootstrapping the ncs scores giving a higher weight to the ncs scores that are closer to the predicted value. Default is FALSE. Experimental, so use with caution.
`contiguize`	logical indicating whether to contiguize the intervals. TRUE will consider all bins for each prediction using the lower and upper endpoints as interval limits to avoid non-contiguous intervals. FALSE will allows for non-contiguous intervals. TRUE guarantees at least appropriate coverage in each bin, but may suffer from over-coverage in certain bins. FALSE will have appropriate coverage in each bin but may have non-contiguous intervals. Default is FALSE.

Value

A tibble with the predicted values, the lower and upper bounds of the prediction intervals. If treat_noncontiguous is 'non_contiguous', the lower and upper bounds are set in a list variable called 'intervals' where all non-contiguous intervals are stored.

Examples

```
library(dplyr)
library(tibble)
x1 <- runif(1000)
x2 <- runif(1000)
y <- rlnorm(1000, meanlog = x1 + x2, sdlog = 0.5)
bin <- cut(y, breaks = quantile(y, probs = seq(0, 1, 1/4)),
include.lowest = TRUE, labels =FALSE)
df <- tibble(x1, x2, y, bin)
df_train <- df %>% slice(1:500)
df_cal <- df %>% slice(501:750)
df_test <- df %>% slice(751:1000)
mod <- lm(log(y) ~ x1 + x2, data=df_train)
calib <- exp(predict(mod, newdata=df_cal))
calib_truth <- df_cal$y
calib_bins <- df_cal$bin
pred_test <- exp(predict(mod, newdata=df_test))

pinterval_bccp(pred = pred_test,
calib = calib,
calib_truth = calib_truth,
calib_bins = calib_bins,
alpha = 0.1,
grid_size = 10000)
```

`pinterval_bootstrap` *Bootstrap prediction intervals*

Description

This function computes bootstrapped prediction intervals with a confidence level of 1-alpha for a vector of (continuous) predicted values using bootstrapped prediction errors. The prediction errors to bootstrap from are computed using either a calibration set with predicted and true values or a set of pre-computed prediction errors from a calibration dataset or other data which the model was not trained on (e.g. OOB errors from a model using bagging). The function returns a tibble containing the predicted values along with the lower and upper bounds of the prediction intervals.

Usage

```
pinterval_bootstrap(
  pred,
  calib = NULL,
  calib_truth = NULL,
  error = NULL,
  error_type = c("raw", "absolute"),
  alpha = 0.1,
  n_bootstraps = 1000,
  lower_bound = NULL,
  upper_bound = NULL
)
```

Arguments

`pred`	Vector of predicted values
`calib`	A numeric vector of predicted values in the calibration partition or a 2 column tibble or matrix with the first column being the predicted values and the second column being the truth values
`calib_truth`	A numeric vector of true values in the calibration partition. Only required if calib is a numeric vector
`error`	An optional numeric vector of pre-computed prediction errors from a calibration partition or other test data. If provided, calib will be ignored
`error_type`	The type of error to use for the prediction intervals. Can be 'raw' or 'absolute'. If 'raw', bootstrapping will be done on the raw prediction errors. If 'absolute', bootstrapping will be done on the absolute prediction errors with random signs. Default is 'raw'
`alpha`	The confidence level for the prediction intervals. Must be a single numeric value between 0 and 1
`n_bootstraps`	The number of bootstraps to perform. Default is 1000
`lower_bound`	Optional minimum value for the prediction intervals. If not provided, the minimum (true) value of the calibration partition will be used

upper_bound	Optional maximum value for the prediction intervals. If not provided, the maximum (true) value of the calibration partition will be used

Value

A tibble with the predicted values, lower bounds, and upper bounds of the prediction intervals

Examples

```
library(dplyr)
library(tibble)
x1 <- runif(1000)
x2 <- runif(1000)
y <- rlnorm(1000, meanlog = x1 + x2, sdlog = 0.5)
df <- tibble(x1, x2, y)
df_train <- df %>% slice(1:500)
df_cal <- df %>% slice(501:750)
df_test <- df %>% slice(751:1000)
mod <- lm(log(y) ~ x1 + x2, data=df_train)
calib <- exp(predict(mod, newdata=df_cal))
calib_truth <- df_cal$y
pred_test <- exp(predict(mod, newdata=df_test))

pinterval_bootstrap(pred = pred_test,
calib = calib,
calib_truth = calib_truth,
error_type = 'raw',
alpha = 0.1,
lower_bound = 0)
```

---

pinterval_conformal *Conformal Prediction Intervals of Continuous Values*

Description

This function calculates conformal prediction intervals with a confidence level of 1-alpha for a vector of (continuous) predicted values using inductive conformal prediction. The intervals are computed using either a calibration set with predicted and true values or a set of pre-computed non-conformity scores from the calibration set. The function returns a tibble containing the predicted values along with the lower and upper bounds of the prediction intervals.

Usage

```
pinterval_conformal(
  pred,
  calib = NULL,
  calib_truth = NULL,
  alpha = 0.1,
  ncs_function = "absolute_error",
  weighted_cp = FALSE,
  ncs = NULL,
  lower_bound = NULL,
  upper_bound = NULL,
  min_step = 0.01,
  grid_size = NULL,
  return_min_q = FALSE
)
```

Arguments

`pred`	Vector of predicted values
`calib`	A numeric vector of predicted values in the calibration partition or a 2 column tibble or matrix with the first column being the predicted values and the second column being the truth values
`calib_truth`	A numeric vector of true values in the calibration partition. Only required if calib is a numeric vector
`alpha`	The confidence level for the prediction intervals. Must be a single numeric value between 0 and 1
`ncs_function`	A function or a character string matching a function that takes two arguments, a vector of predicted values and a vector of true values, in that order. The function should return a numeric vector of nonconformity scores. Default is 'absolute_error' which returns the absolute difference between the predicted and true values.
`weighted_cp`	Logical. If TRUE, the function will use weighted conformal prediction. Default is FALSE. Experimental, use with caution.
`ncs`	A numeric vector of pre-computed nonconformity scores from a calibration partition. If provided, calib will be ignored
`lower_bound`	Optional minimum value for the prediction intervals. If not provided, the minimum (true) value of the calibration partition will be used
`upper_bound`	Optional maximum value for the prediction intervals. If not provided, the maximum (true) value of the calibration partition will be used
`min_step`	The minimum step size for the grid search. Default is 0.01. Useful to change if predictions are made on a discrete grid or if the resolution of the interval is too coarse or too fine.
`grid_size`	Alternative to min_step, the number of points to use in the grid search between the lower and upper bound. If provided, min_step will be ignored.
`return_min_q`	Logical. If TRUE, the function will return the minimum quantile of the nonconformity scores for each predicted value. Default is FALSE. Primarily used for debugging purposes.

Value

A tibble with the predicted values and the lower and upper bounds of the prediction intervals.

Examples

```
library(dplyr)
library(tibble)
x1 <- runif(1000)
x2 <- runif(1000)
y <- rlnorm(1000, meanlog = x1 + x2, sdlog = 0.5)
df <- tibble(x1, x2, y)
df_train <- df %>% slice(1:500)
df_cal <- df %>% slice(501:750)
df_test <- df %>% slice(751:1000)
mod <- lm(log(y) ~ x1 + x2, data=df_train)
calib <- exp(predict(mod, newdata=df_cal))
calib_truth <- df_cal$y
pred_test <- exp(predict(mod, newdata=df_test))

pinterval_conformal(pred_test,
calib = calib,
calib_truth = calib_truth,
alpha = 0.1,
lower_bound = 0,
grid_size = 10000)
```

`pinterval_parametric` *Parametric prediction intervals for continuous predictions*

Description

This function computes parametric prediction intervals with a confidence level of 1-alpha for a vector of (continuous) predicted values using a user specified parametric distribution and parameters. The distribution can be any distribution available in R or a user defined distribution as long as a quantile function is available. The parameters should be estimated on calibration data. The prediction intervals are calculated as the quantiles of the distribution at the specified confidence level.

Usage

```
pinterval_parametric(
  pred,
  dist = c("norm", "lnorm", "pois", "nbinom", "gamma", "logis", "beta"),
  pars = list(),
  alpha = 0.1,
  lower_bound = NULL,
  upper_bound = NULL
)
```

Arguments

`pred`	Vector of predicted values
`dist`	Distribution to use for the prediction intervals. Can be a character string matching any available distribution in R or a function representing a distribution. If a function is provided, it must be a quantile function (e.g. qnorm, qgamma, etc.)
`pars`	List of named parameters for the distribution for each prediction. See details for more information.
`alpha`	The confidence level for the prediction intervals. Must be a single numeric value between 0 and 1
`lower_bound`	Optional minimum value for the prediction intervals. If not provided, the minimum (true) value of the calibration partition will be used
`upper_bound`	Optional maximum value for the prediction intervals. If not provided, the maximum (true) value of the calibration partition will be used

Details

The distributions are not limited to the standard distributions available in R. Any distribution can be used as long as a quantile function is available. Users may create their own distribution functions and plug in the resulting quantile function or create compositie or mixture distributions using for instance the package 'mistr' and plug in the resulting quantile function.

The list of parameters should be constructed such that when the distribution function is called with the parameters, it returns a vector of the same length as the predictions. In most cases the parameters should ensure that the predicted value corresponds to the mean, median, or mode of the resulting distribution. Parameters relating to the prediction error should be estimated on calibration data. For example, if normal prediction intervals are desired, the mean parameter should be the predicted value and the standard deviation parameter should be the estimated standard deviation of the prediction errors in the calibration set. If the distribution is a negative binomial distribution with a fixed size parameter, the size parameter should be estimated on the calibration data and the mu parameter should be the predicted value.

Value

A tibble with the predicted values and the lower and upper bounds of the prediction intervals

Examples

```
library(dplyr)
library(tibble)
x1 <- runif(1000)
x2 <- runif(1000)
y <- rlnorm(1000, meanlog = x1 + x2, sdlog = 0.5)
df <- tibble(x1, x2, y)
df_train <- df %>% slice(1:500)
df_cal <- df %>% slice(501:750)
df_test <- df %>% slice(751:1000)
mod <- lm(log(y) ~ x1 + x2, data=df_train)
calib <- exp(predict(mod, newdata=df_cal))
calib_truth <- df_cal$y
```

```
pred_test <- exp(predict(mod, newdata=df_test))

# Normal prediction intervals
pinterval_parametric(pred = pred_test,
dist = 'norm',
pars = list(mean = pred_test,
sd = sqrt(mean((calib - calib_truth)^2))))

# Log-normal prediction intervals
pinterval_parametric(pred = pred_test,
dist = 'lnorm',
pars = list(meanlog = pred_test,
sdlog = sqrt(mean((log(calib) - log(calib_truth))^2))))
```

`weights_calculator` *Weights calculator for weighted conformal prediction*

Description

Weights calculator for weighted conformal prediction

Usage

```
weights_calculator(y_hat, calib)
```

Arguments

`y_hat`	Predicted value
`calib`	a vector of true values of the calibration partition

Index

abs_error, 2

bin_chopper, 3
bindividual_alpha, 2
bootstrap_inner, 3

contiguize_intervals, 4

flatten_cp_bin_intervals, 4

grid_finder, 5
grid_inner, 6

minq_to_alpha, 6

pinterval_bcbootstrap, 7
pinterval_bccp, 8
pinterval_bootstrap, 11
pinterval_conformal, 12
pinterval_parametric, 14

weights_calculator, 16