Package ‘predictset’

March 19, 2026

Title Conformal Prediction and Uncertainty Quantification

Version 0.3.0

Description Implements conformal prediction methods for constructing prediction intervals (regression) and prediction sets (classification) with finite-sample coverage guarantees. Methods include split conformal, 'CV+' and 'Jackknife+' (Barber et al. 2021) <doi:10.1214/20-AOS1965>, 'Conformalized Quantile Regression' (Romano et al. 2019) <doi:10.48550/arXiv.1905.03222>, 'Adaptive Prediction Sets' (Romano, Sesia, Candes 2020) <doi:10.48550/arXiv.2006.02544>, 'Regularized Adaptive Prediction Sets' (Angelopoulos et al. 2021) <doi:10.48550/arXiv.2009.14193>, Mondrian conformal prediction for group-conditional coverage (Vovk et al. 2005), weighted conformal prediction for covariate shift (Tibshirani et al. 2019), and adaptive conformal inference for sequential prediction (Gibbs and Candes 2021). All methods are distribution-free and provide calibrated uncertainty quantification without parametric assumptions. Works with any model that can produce predictions from new data, including 'lm', 'glm', 'ranger', 'xgboost', and custom user-defined models.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL `https://github.com/charlescoverdale/predictset`

BugReports `https://github.com/charlescoverdale/predictset/issues`

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports cli (>= 3.6.0), grDevices, graphics, stats

Suggests testthat (>= 3.0.0), ranger, ggplot2, knitr, rmarkdown, parsnip (>= 1.0.0), probably, rsample, workflows

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Charles Coverdale [aut, cre, cph]

Maintainer Charles Coverdale <charlesfcoverdale@gmail.com>

Repository CRAN

Date/Publication 2026-03-19 14:50:15 UTC

Contents

conformal_aci 3
conformal_aps 4
conformal_class_split 6
conformal_compare 7
conformal_cqr 9
conformal_cv 10
conformal_jackknife 12
conformal_lac 14
conformal_mondrian 15
conformal_mondrian_class 17
conformal_pvalue 18
conformal_raps 19
conformal_split 21
conformal_weighted 23
coverage 24
coverage_by_bin 25
coverage_by_group 26
interval_width 27
make_model 28
plot.predictset_aci 29
plot.predictset_class 29
plot.predictset_reg 30
predict.predictset_class 31
predict.predictset_reg 32
print.predictset_aci 33
print.predictset_class 33
print.predictset_model 34
print.predictset_reg 34
set_size 35
summary.predictset_class 36
summary.predictset_reg 36

Index **37**

`conformal_aci` *Adaptive Conformal Inference*

Description

Implements basic Adaptive Conformal Inference (ACI) for sequential prediction. The miscoverage level alpha is adjusted online based on whether previous predictions covered the true values, maintaining long-run coverage even under distribution shift.

Usage

```
conformal_aci(y_pred, y_true, alpha = 0.1, gamma = 0.005)
```

Arguments

`y_pred`	A numeric vector of point predictions (sequential).
`y_true`	A numeric vector of true values (sequential).
`alpha`	Target miscoverage level. Default `0.10`.
`gamma`	Learning rate for alpha adjustment. Default `0.005`. Larger values adapt faster but are less stable.

Details

ACI provides asymptotic coverage guarantees under distribution drift, not the finite-sample guarantees of split conformal prediction. The long-run average coverage converges to $1 - \alpha$ as the sequence length grows (Gibbs and Candes, 2021).

Value

A list with components:

lower Numeric vector of lower bounds.

upper Numeric vector of upper bounds.

covered Logical vector indicating whether each interval covered the true value.

alphas Numeric vector of the adapted alpha values at each step.

coverage Overall empirical coverage.

References

Gibbs, I. and Candes, E. (2021). Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34.

See Also

Other regression methods: `conformal_cqr()`, `conformal_cv()`, `conformal_jackknife()`, `conformal_mondrian()`, `conformal_split()`, `conformal_weighted()`

Examples

```
set.seed(42)
n <- 200
y_true <- cumsum(rnorm(n, sd = 0.1)) + rnorm(n)
y_pred <- c(0, y_true[-n])  # naive lag-1 prediction

result <- conformal_aci(y_pred, y_true, alpha = 0.10, gamma = 0.01)
print(result$coverage)
```

`conformal_aps` *Adaptive Prediction Sets*

Description

Constructs prediction sets using the Adaptive Prediction Sets (APS) method of Romano, Sesia, and Candes (2020). Classes are included in order of decreasing predicted probability until the cumulative probability exceeds the conformal threshold.

Usage

```
conformal_aps(
  x,
  y,
  model,
  x_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  randomize = FALSE,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A factor (or character/integer vector coerced to factor) of class labels.
`model`	A `make_model()` specification with `type = "classification"`, or a fitted model object that produces class probabilities.
`x_new`	A numeric matrix or data frame of new predictor variables.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction sets.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`randomize`	Logical. If `TRUE`, uses randomized scores for exact coverage (but prediction sets become stochastic). Default `FALSE`.
`seed`	Optional random seed.

Details

When `randomize` = `FALSE` (the default), this implementation uses the deterministic variant of APS, which provides conservative coverage (at least $1-\alpha$). The randomized variant (`randomize` = `TRUE`) achieves exact $1-\alpha$ coverage but produces non-reproducible prediction sets.

Value

A `predictset_class` object. See `conformal_lac()` for details. The `method` component is `"aps"`.

References

Romano, Y., Sesia, M. and Candes, E.J. (2020). Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33. doi:10.48550/arXiv.2006.02544

See Also

Other classification methods: `conformal_lac()`, `conformal_mondrian_class()`, `conformal_raps()`

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(sample(c("A", "B", "C"), n, replace = TRUE))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                 family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = p / 2, B = p / 2, C = 1 - p)
  },
  type = "classification"
)


result <- conformal_aps(x, y, model = clf, x_new = x_new)
print(result)
```

conformal_class_split *Split Conformal Prediction Sets for Classification*

Description

[Deprecated] `conformal_class_split()` is identical to `conformal_lac()` and is deprecated. Use `conformal_lac()` instead.

Usage

```
conformal_class_split(
  x,
  y,
  model,
  x_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A factor (or character/integer vector coerced to factor) of class labels.
`model`	A `make_model()` specification with `type = "classification"`, or a fitted model object that produces class probabilities.
`x_new`	A numeric matrix or data frame of new predictor variables.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction sets.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`seed`	Optional random seed.

Value

A `predictset_class` object. See `conformal_lac()` for details.

References

Sadinle, M., Lei, J. and Wasserman, L. (2019). Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525), 223-234. doi:10.1080/01621459.2017.1395341

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(ifelse(x[,1] + x[,2] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) {
    df <- data.frame(y = y, x)
    glm(y ~ ., data = df, family = "binomial")
  },
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)


suppressWarnings(
  result <- conformal_class_split(x, y, model = clf, x_new = x_new)
)
```

`conformal_compare` *Compare Conformal Prediction Methods*

Description

Runs multiple conformal prediction methods on the same data and returns a comparison data frame with coverage, interval width, and computation time for each method.

Usage

```
conformal_compare(
  x,
  y,
  model,
  x_new,
  y_new,
  methods = c("split", "cv"),
  alpha = 0.1,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model`	A fitted model object, a `make_model()` specification, or a formula.
`x_new`	A numeric matrix or data frame of new predictor variables.
`y_new`	A numeric vector of true response values for `x_new`, used to compute empirical coverage.
`methods`	Character vector of method names to compare. Default `c("split", "cv")`. Available methods: `"split"`, `"cv"`, `"jackknife"`, `"jackknife_basic"`.
`alpha`	Miscoverage level. Default `0.10`.
`seed`	Optional random seed.

Value

A `predictset_compare` object (a data frame) with columns:

method Character. The method name.

coverage Numeric. Empirical coverage on `y_new`.

mean_width Numeric. Mean interval width.

median_width Numeric. Median interval width.

time_seconds Numeric. Elapsed time in seconds.

See Also

Other diagnostics: `conformal_pvalue()`, `coverage()`, `coverage_by_bin()`, `coverage_by_group()`, `interval_width()`, `set_size()`

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(100 * 3), ncol = 3)
y_new <- x_new[, 1] * 2 + rnorm(100)


comp <- conformal_compare(x, y, model = y ~ ., x_new = x_new,
                           y_new = y_new)
print(comp)
```

`conformal_cqr` *Conformalized Quantile Regression*

Description

Constructs prediction intervals using Conformalized Quantile Regression (Romano et al. 2019). Requires two models: one for the lower quantile and one for the upper quantile. The conformal step adjusts these quantile predictions to achieve valid coverage.

Usage

```
conformal_cqr(
  x,
  y,
  model_lower,
  model_upper,
  x_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  quantiles = c(0.05, 0.95),
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model_lower`	A `make_model()` specification for the lower quantile model.
`model_upper`	A `make_model()` specification for the upper quantile model.
`x_new`	A numeric matrix or data frame of new predictor variables.
`alpha`	Miscoverage level. Default `0.10`.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`quantiles`	The target quantiles. Default `c(0.05, 0.95)`.
`seed`	Optional random seed.

Details

Interval quality depends on the underlying quantile models. Poorly calibrated quantile models produce valid but potentially wide intervals. For best results, use proper quantile regression models (e.g. `quantreg::rq()`) rather than shifted mean predictions.

Value

A `predictset_reg` object. See `conformal_split()` for details. The `method` component is `"cqr"`.

References

Romano, Y., Patterson, E. and Candes, E.J. (2019). Conformalized quantile regression. *Advances in Neural Information Processing Systems*, 32. doi:10.48550/arXiv.1905.03222

See Also

Other regression methods: `conformal_aci()`, `conformal_cv()`, `conformal_jackknife()`, `conformal_mondrian()`, `conformal_split()`, `conformal_weighted()`

Examples

```
set.seed(42)
n <- 200
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(20 * 3), ncol = 3)

# Approximating quantile regression with shifted linear models.
# In practice, use quantile regression models, e.g.:
#   quantreg::rq(y ~ ., data = df, tau = 0.05)
model_lo <- make_model(
  train_fun = function(x, y) lm(y ~ ., data = data.frame(y = y, x)),
  predict_fun = function(obj, x_new) {
    predict(obj, newdata = as.data.frame(x_new)) - 1.5
  },
  type = "regression"
)
model_hi <- make_model(
  train_fun = function(x, y) lm(y ~ ., data = data.frame(y = y, x)),
  predict_fun = function(obj, x_new) {
    predict(obj, newdata = as.data.frame(x_new)) + 1.5
  },
  type = "regression"
)

result <- conformal_cqr(x, y, model_lo, model_hi, x_new = x_new)
print(result)
```

conformal_cv *CV+ Conformal Prediction Intervals*

Description

Constructs prediction intervals using the CV+ method of Barber et al. (2021). Cross-validation residuals and fold-specific models are used to form observation-specific prediction intervals with finite-sample coverage guarantees.

Usage

```
conformal_cv(
  x,
  y,
  model,
  x_new = NULL,
  alpha = 0.1,
  n_folds = 10,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model`	A fitted model object (e.g., from `lm()`), a `make_model()` specification, or a formula (which will fit a linear model).
`x_new`	A numeric matrix or data frame of new predictor variables. If `NULL`, intervals are computed for the training data using leave-one-fold-out predictions. Note: when `x_new = NULL`, prediction intervals for training observations use a self-consistent approximation. For exact CV+ intervals on new data, provide `x_new`.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction intervals.
`n_folds`	Number of cross-validation folds. Default `10`.
`verbose`	Logical. If `TRUE`, shows a progress bar during fold fitting. Default `FALSE`.
`seed`	Optional random seed for reproducible data splitting.

Details

Unlike basic CV conformal prediction (which computes a single quantile of CV residuals), CV+ constructs intervals that vary per test point. For each test point, every training observation contributes a lower and upper value based on the fold model that excluded that observation, evaluated at the test point, plus or minus the leave-fold-out residual for that observation. The interval bounds are then taken as quantiles of these per-observation values.

The CV+ theoretical coverage guarantee is $1 - 2\alpha$, not $1 - \alpha$ (Barber et al. 2021, Theorem 2). This is weaker than split conformal's $1 - \alpha$ guarantee. In practice, CV+ coverage is typically much closer to $1 - \alpha$.

Value

A `predictset_reg` object. See `conformal_split()` for details. The `method` component is `"cv_plus"`. The object also stores `fold_models` (list of K fitted models), `fold_ids` (integer vector mapping each observation to its fold), and `residuals` (leave-fold-out absolute residuals), which are needed by the `predict()` method to compute CV+ intervals for new data.

References

Barber, R.F., Candes, E.J., Ramdas, A. and Tibshirani, R.J. (2021). Predictive inference with the Jackknife+. *Annals of Statistics*, 49(1), 486-507. doi:10.1214/20AOS1965

See Also

Other regression methods: `conformal_aci()`, `conformal_cqr()`, `conformal_jackknife()`, `conformal_mondrian()`, `conformal_split()`, `conformal_weighted()`

Examples

```
set.seed(42)
n <- 200
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(20 * 3), ncol = 3)


result <- conformal_cv(x, y, model = y ~ ., x_new = x_new, n_folds = 5)
print(result)
```

`conformal_jackknife` *Jackknife+ Conformal Prediction Intervals*

Description

Constructs prediction intervals using the Jackknife+ method of Barber et al. (2021). Uses leave-one-out models to form prediction intervals with finite-sample coverage guarantees.

Usage

```
conformal_jackknife(
  x,
  y,
  model,
  x_new = NULL,
  alpha = 0.1,
  plus = TRUE,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model`	A fitted model object (e.g., from `lm()`), a `make_model()` specification, or a formula (which will fit a linear model).
`x_new`	A numeric matrix or data frame of new predictor variables. If `NULL`, intervals are computed for the training data.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction intervals.
`plus`	Logical. If `TRUE` (default), uses the Jackknife+ method. If `FALSE`, uses the basic jackknife.
`verbose`	Logical. If `TRUE`, shows a progress bar during LOO fitting. Default `FALSE`.
`seed`	Optional random seed for reproducible data splitting.

Details

The Jackknife+ method fits n leave-one-out models and uses each model's prediction at the test point, shifted by the corresponding LOO residual, to construct the interval. This is distinct from basic jackknife, which centres a single full-model prediction and adds a quantile of the LOO residuals.

The Jackknife+ theoretical coverage guarantee is $1-2\alpha$, not $1-\alpha$ (Barber et al. 2021, Theorem 1). This is weaker than split conformal's $1-\alpha$ guarantee. In practice, Jackknife+ coverage is typically much closer to $1-\alpha$.

Value

A `predictset_reg` object. See `conformal_split()` for details. The `method` component is `"jackknife_plus"` or `"jackknife"`. Additional components include `loo_models` (list of n leave-one-out fitted models) and `loo_residuals` (numeric vector of LOO absolute residuals).

References

Barber, R.F., Candes, E.J., Ramdas, A. and Tibshirani, R.J. (2021). Predictive inference with the jackknife+. *Annals of Statistics*, 49(1), 486–507.

See Also

Other regression methods: `conformal_aci()`, `conformal_cqr()`, `conformal_cv()`, `conformal_mondrian()`, `conformal_split()`, `conformal_weighted()`

Examples

```
set.seed(42)
n <- 50
x <- matrix(rnorm(n * 2), ncol = 2)
y <- x[, 1] + rnorm(n)
x_new <- matrix(rnorm(10 * 2), ncol = 2)
```

```
result <- conformal_jackknife(x, y, model = y ~ ., x_new = x_new)
print(result)
```

conformal_lac *Least Ambiguous Classifier Prediction Sets*

Description

Constructs prediction sets using the Least Ambiguous Classifier (LAC) method. Includes all classes whose predicted probability exceeds `1 - q`, where `q` is the conformal quantile of `1 - p(true class)` scores.

Usage

```
conformal_lac(x, y, model, x_new, alpha = 0.1, cal_fraction = 0.5, seed = NULL)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A factor (or character/integer vector coerced to factor) of class labels.
`model`	A `make_model()` specification with `type = "classification"`, or a fitted model object that produces class probabilities.
`x_new`	A numeric matrix or data frame of new predictor variables.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction sets.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`seed`	Optional random seed.

Value

A `predictset_class` object with components:

sets A list of character vectors, one per new observation.

probs A list of named numeric vectors with predicted probabilities for included classes.

alpha The miscoverage level used.

method Character string `"lac"`.

scores Numeric vector of calibration scores.

quantile The conformal quantile used.

classes Character vector of all class labels.

n_cal Number of calibration observations.

n_train Number of training observations.

fitted_model The fitted model object.

model The `predictset_model` specification.

References

Sadinle, M., Lei, J. and Wasserman, L. (2019). Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525), 223-234. doi:10.1080/01621459.2017.1395341

See Also

Other classification methods: `conformal_aps()`, `conformal_mondrian_class()`, `conformal_raps()`

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(ifelse(x[,1] + x[,2] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                 family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)

result <- conformal_lac(x, y, model = clf, x_new = x_new)
print(result)
```

`conformal_mondrian` *Mondrian Conformal Prediction Intervals (Group-Conditional)*

Description

Constructs prediction intervals with group-conditional coverage guarantees. Instead of a single conformal quantile, a separate quantile is computed for each group, ensuring coverage within each subgroup (e.g. by gender, region, or model type).

Usage

```
conformal_mondrian(
  x,
  y,
  model,
  x_new,
  groups,
  groups_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  seed = NULL
)
```

Arguments

x	A numeric matrix or data frame of predictor variables.
y	A numeric vector of response values.
model	A fitted model object, a make_model() specification, or a formula.
x_new	A numeric matrix or data frame of new predictor variables.
groups	A factor or character vector of group labels for each observation in x, with length equal to nrow(x).
groups_new	A factor or character vector of group labels for each observation in x_new, with length equal to nrow(x_new).
alpha	Miscoverage level. Default 0.10.
cal_fraction	Fraction of data used for calibration. Default 0.5.
seed	Optional random seed.

Value

A predictset_reg object. See conformal_split() for details. The method component is "mondrian". Additional components include groups_new (the group labels for new data) and group_quantiles (named numeric vector of per-group conformal quantiles).

References

Vovk, V., Gammerman, A. and Shafer, G. (2005). *Algorithmic Learning in a Random World.* Springer.

See Also

Other regression methods: conformal_aci(), conformal_cqr(), conformal_cv(), conformal_jackknife(), conformal_split(), conformal_weighted()

Examples

```
set.seed(42)
n <- 400
x <- matrix(rnorm(n * 3), ncol = 3)
groups <- factor(ifelse(x[, 1] > 0, "high", "low"))
y <- x[, 1] * 2 + ifelse(groups == "high", 2, 0.5) * rnorm(n)
x_new <- matrix(rnorm(50 * 3), ncol = 3)
groups_new <- factor(ifelse(x_new[, 1] > 0, "high", "low"))
```

```
result <- conformal_mondrian(x, y, model = y ~ ., x_new = x_new,
                             groups = groups, groups_new = groups_new)
print(result)
```

---

conformal_mondrian_class *Mondrian Conformal Prediction Sets for Classification*

Description

Constructs prediction sets with group-conditional coverage guarantees for classification. Uses LAC-style scoring with per-group conformal quantiles.

Usage

```
conformal_mondrian_class(
  x,
  y,
  model,
  x_new,
  groups,
  groups_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  seed = NULL
)
```

Arguments

x	A numeric matrix or data frame of predictor variables.
y	A factor (or character/integer vector coerced to factor) of class labels.
model	A `make_model()` specification with `type = "classification"`, or a fitted model object.
x_new	A numeric matrix or data frame of new predictor variables.
groups	A factor or character vector of group labels for each observation in `x`.
groups_new	A factor or character vector of group labels for each observation in `x_new`.
alpha	Miscoverage level. Default `0.10`.
cal_fraction	Fraction of data used for calibration. Default `0.5`.
seed	Optional random seed.

Value

A `predictset_class` object. See `conformal_lac()` for details. The `method` component is `"mondrian"`. Additional components include `groups_new` and `group_quantiles`.

See Also

Other classification methods: `conformal_aps()`, `conformal_lac()`, `conformal_raps()`

Examples

```
set.seed(42)
n <- 400
x <- matrix(rnorm(n * 4), ncol = 4)
groups <- factor(ifelse(x[, 1] > 0, "high", "low"))
y <- factor(ifelse(x[,1] + x[,2] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)
groups_new <- factor(ifelse(x_new[, 1] > 0, "high", "low"))

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                  family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)


result <- conformal_mondrian_class(x, y, model = clf, x_new = x_new,
                                    groups = groups, groups_new = groups_new)
print(result)
```

---

conformal_pvalue *Conformal P-Values*

Description

Computes conformal p-values for new observations given calibration nonconformity scores. The p-value indicates how conforming a new observation is relative to the calibration set.

Usage

```
conformal_pvalue(scores, new_scores)
```

Arguments

`scores`	A numeric vector of calibration nonconformity scores.
`new_scores`	A numeric vector of nonconformity scores for new observations.

Value

A numeric vector of p-values, one per element of `new_scores`. Each p-value is in (0, 1].

See Also

Other diagnostics: `conformal_compare()`, `coverage()`, `coverage_by_bin()`, `coverage_by_group()`, `interval_width()`, `set_size()`

Examples

```
# Calibration scores from a conformal split
set.seed(42)
cal_scores <- abs(rnorm(100))
new_scores <- abs(rnorm(5))

pvals <- conformal_pvalue(cal_scores, new_scores)
print(pvals)
```

`conformal_raps` *Regularized Adaptive Prediction Sets*

Description

Constructs prediction sets using the Regularized Adaptive Prediction Sets (RAPS) method of Angelopoulos et al. (2021). Extends APS with a regularization penalty that encourages smaller prediction sets.

Usage

```
conformal_raps(
  x,
  y,
  model,
  x_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  k_reg = 1,
  lambda = 0.01,
  randomize = FALSE,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A factor (or character/integer vector coerced to factor) of class labels.
`model`	A `make_model()` specification with `type = "classification"`, or a fitted model object that produces class probabilities.
`x_new`	A numeric matrix or data frame of new predictor variables.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction sets.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`k_reg`	Regularization parameter controlling the number of classes exempt from the penalty. Default `1` (only the top class is unpenalized).
`lambda`	Regularization strength. Default `0.01`. Larger values produce smaller prediction sets at the potential cost of coverage.
`randomize`	Logical. If `TRUE`, uses randomized scores for exact coverage (but prediction sets become stochastic). Default `FALSE`.
`seed`	Optional random seed.

Value

A `predictset_class` object. See `conformal_lac()` for details. The `method` component is `"raps"`.

References

Angelopoulos, A.N., Bates, S., Malik, J. and Jordan, M.I. (2021). Uncertainty sets for image classifiers using conformal prediction. *International Conference on Learning Representations*. doi:10.48550/arXiv.2009.14193

See Also

Other classification methods: `conformal_aps()`, `conformal_lac()`, `conformal_mondrian_class()`

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(sample(c("A", "B", "C"), n, replace = TRUE))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                  family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = p / 2, B = p / 2, C = 1 - p)
  },
  type = "classification"
)


result <- conformal_raps(x, y, model = clf, x_new = x_new,
                         k_reg = 1, lambda = 0.01)
print(result)
```

`conformal_split` *Split Conformal Prediction Intervals*

Description

Constructs prediction intervals using split conformal inference. The data is split into training and calibration sets; nonconformity scores are computed on the calibration set and used to form intervals on new data.

Usage

```
conformal_split(
  x,
  y,
  model,
  x_new,
  alpha = 0.1,
  cal_fraction = 0.5,
  score_type = c("absolute", "normalized"),
  scale_model = NULL,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model`	A fitted model object (e.g., from `lm()`), a `make_model()` specification, or a formula (which will fit a linear model).
`x_new`	A numeric matrix or data frame of new predictor variables for which to compute prediction intervals.
`alpha`	Miscoverage level. Default `0.10` gives 90 percent prediction intervals.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`score_type`	Type of nonconformity score. `"absolute"` (default) uses absolute residuals and produces constant-width intervals. `"normalized"` divides residuals by a local scale estimate from `scale_model`, producing locally-adaptive interval widths.

`scale_model`	A `make_model()` specification for predicting absolute residuals (used only when `score_type = "normalized"`). Must return positive predictions. If `NULL` and `score_type = "normalized"`, a default model is fitted using `lm()` on absolute residuals.
`seed`	Optional random seed for reproducible data splitting.

Value

A `predictset_reg` object (a list) with components:

pred Numeric vector of point predictions for `x_new`.

lower Numeric vector of lower bounds.

upper Numeric vector of upper bounds.

alpha The miscoverage level used.

method Character string `"split"`.

scores Numeric vector of calibration nonconformity scores.

quantile The conformal quantile used to form intervals.

n_cal Number of calibration observations.

n_train Number of training observations.

fitted_model The fitted model object.

model The `predictset_model` specification.

References

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R.J. and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523), 1094-1111. doi:10.1080/01621459.2017.1307116

See Also

Other regression methods: `conformal_aci()`, `conformal_cqr()`, `conformal_cv()`, `conformal_jackknife()`, `conformal_mondrian()`, `conformal_weighted()`

Examples

```
set.seed(42)
n <- 200
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(50 * 3), ncol = 3)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
print(result)
```

conformal_weighted *Weighted Conformal Prediction Intervals*

Description

Constructs prediction intervals using weighted split conformal inference, designed for settings with covariate shift where calibration and test data may have different distributions. Importance weights re-weight the calibration scores to account for this shift.

Usage

```
conformal_weighted(
  x,
  y,
  model,
  x_new,
  weights = NULL,
  alpha = 0.1,
  cal_fraction = 0.5,
  seed = NULL
)
```

Arguments

`x`	A numeric matrix or data frame of predictor variables.
`y`	A numeric vector of response values.
`model`	A fitted model object, a `make_model()` specification, or a formula.
`x_new`	A numeric matrix or data frame of new predictor variables.
`weights`	A numeric vector of importance weights for each observation in `x`, with length equal to `nrow(x)`. Weights must be non-negative. If `NULL`, uniform weights are used (equivalent to standard split conformal).
`alpha`	Miscoverage level. Default `0.10`.
`cal_fraction`	Fraction of data used for calibration. Default `0.5`.
`seed`	Optional random seed.

Details

The test-point weight w_{n+1} is set to the mean of calibration weights, following standard practice when the true test weight is unknown. See Tibshirani et al. (2019), Equation 5.

Value

A `predictset_reg` object. See `conformal_split()` for details. The `method` component is `"weighted"`.

References

Tibshirani, R.J., Barber, R.F., Candes, E.J. and Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in Neural Information Processing Systems*, 32.

See Also

Other regression methods: `conformal_aci()`, `conformal_cqr()`, `conformal_cv()`, `conformal_jackknife()`, `conformal_mondrian()`, `conformal_split()`

Examples

```
set.seed(42)
n <- 400
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(50 * 3, mean = 1), ncol = 3)
weights <- rep(1, n)


result <- conformal_weighted(x, y, model = y ~ ., x_new = x_new,
                              weights = weights)
print(result)
```

`coverage` *Empirical Coverage Rate*

Description

Computes the fraction of true values that fall within the prediction intervals (regression) or prediction sets (classification).

Usage

```
coverage(object, y_true)

## S3 method for class 'predictset_reg'
coverage(object, y_true)

## S3 method for class 'predictset_class'
coverage(object, y_true)
```

Arguments

`object`	A `predictset_reg` or `predictset_class` object.
`y_true`	A numeric vector (regression) or factor/character vector (classification) of true values, with the same length as the number of predictions.

Value

A single numeric value between 0 and 1 representing the empirical coverage rate.

See Also

Other diagnostics: `conformal_compare()`, `conformal_pvalue()`, `coverage_by_bin()`, `coverage_by_group()`, `interval_width()`, `set_size()`

Examples

```
set.seed(42)
n <- 500
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(100 * 3), ncol = 3)
y_new <- x_new[, 1] * 2 + rnorm(100)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
coverage(result, y_new)
```

`coverage_by_bin` *Empirical Coverage by Prediction Bin*

Description

Bins predictions into quantile-based groups and computes coverage within each bin. Useful for detecting systematic under- or over-coverage as a function of predicted value.

Usage

```
coverage_by_bin(object, y_true, bins = 10)
```

Arguments

`object`	A `predictset_reg` object.
`y_true`	A numeric vector of true response values.
`bins`	Number of bins. Default `10`.

Value

A data frame with columns `bin`, `coverage`, `n`, and `mean_width`.

See Also

Other diagnostics: `conformal_compare()`, `conformal_pvalue()`, `coverage()`, `coverage_by_group()`, `interval_width()`, `set_size()`

Examples

```
set.seed(42)
n <- 500
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(200 * 3), ncol = 3)
y_new <- x_new[, 1] * 2 + rnorm(200)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
coverage_by_bin(result, y_new, bins = 5)
```

`coverage_by_group` *Empirical Coverage by Group*

Description

Computes empirical coverage within each group, useful for diagnosing conditional coverage violations.

Usage

```
coverage_by_group(object, y_true, groups)
```

Arguments

`object`	A `predictset_reg` or `predictset_class` object.
`y_true`	True values (numeric for regression, factor/character for classification).
`groups`	A factor or character vector of group labels with the same length as the number of predictions.

Value

A data frame with columns `group`, `coverage`, `n`, and `target`.

See Also

Other diagnostics: `conformal_compare()`, `conformal_pvalue()`, `coverage()`, `coverage_by_bin()`, `interval_width()`, `set_size()`

Examples

```
set.seed(42)
n <- 500
x <- matrix(rnorm(n * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(n)
x_new <- matrix(rnorm(200 * 3), ncol = 3)
y_new <- x_new[, 1] * 2 + rnorm(200)
```

```
groups <- factor(ifelse(x_new[, 1] > 0, "high", "low"))

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
coverage_by_group(result, y_new, groups)
```

`interval_width` *Prediction Interval Widths*

Description

Returns the width of each prediction interval.

Usage

```
interval_width(object)
```

Arguments

`object`	A `predictset_reg` object.

Value

A numeric vector of interval widths.

See Also

Other diagnostics: `conformal_compare()`, `conformal_pvalue()`, `coverage()`, `coverage_by_bin()`, `coverage_by_group()`, `set_size()`

Examples

```
set.seed(42)
x <- matrix(rnorm(200 * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(200)
x_new <- matrix(rnorm(50 * 3), ncol = 3)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
widths <- interval_width(result)
summary(widths)
```

make_model *Create a Model Specification for Conformal Prediction*

Description

Defines how to train a model and generate predictions, allowing any model to be used with conformal prediction methods.

Usage

```
make_model(train_fun, predict_fun, type = c("regression", "classification"))
```

Arguments

`train_fun`	A function with signature `function(x, y)` that takes a numeric matrix `x` and response `y` (numeric for regression, factor for classification) and returns a fitted model object.
`predict_fun`	A function with signature `function(object, x_new)` that takes a fitted model object and a numeric matrix `x_new` and returns predictions. For regression, must return a numeric vector. For classification, must return a probability matrix with columns named by class labels.
`type`	Character string, either `"regression"` or `"classification"`.

Value

A `predictset_model` object (a list with components `train_fun`, `predict_fun`, and `type`).

Examples

```
reg_model <- make_model(
  train_fun = function(x, y) lm(y ~ ., data = data.frame(y = y, x)),
  predict_fun = function(object, x_new) {
    predict(object, newdata = as.data.frame(x_new))
  },
  type = "regression"
)
```

`plot.predictset_aci` *Plot Method for ACI Objects*

Description

Creates a two-panel base R plot. The top panel shows prediction intervals over time; the bottom panel shows the adaptive alpha trace.

Usage

```
## S3 method for class 'predictset_aci'
plot(x, max_points = 500, ...)
```

Arguments

`x`	A `predictset_aci` object.
`max_points`	Maximum number of points to display. Default `500`.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

Examples

```
set.seed(42)
n <- 100
y_true <- cumsum(rnorm(n, sd = 0.1)) + rnorm(n)
y_pred <- c(0, y_true[-n])


result <- conformal_aci(y_pred, y_true, alpha = 0.10, gamma = 0.01)
plot(result)
```

`plot.predictset_class` *Plot Method for Classification Conformal Objects*

Description

Creates a barplot showing the distribution of prediction set sizes.

Usage

```
## S3 method for class 'predictset_class'
plot(x, ...)
```

Arguments

`x`	A `predictset_class` object.
`...`	Additional arguments passed to `barplot()`.

Value

The input object, invisibly.

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(ifelse(x[,1] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                  family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)

result <- conformal_lac(x, y, model = clf, x_new = x_new)
plot(result)
```

`plot.predictset_reg` *Plot Method for Regression Conformal Objects*

Description

Creates a base R plot showing prediction intervals. Points are ordered by predicted value, with intervals shown as vertical segments.

Usage

```
## S3 method for class 'predictset_reg'
plot(x, max_points = 200, ...)
```

Arguments

`x`	A `predictset_reg` object.
`max_points`	Maximum number of points to display. Default `200`. If there are more predictions, a random subset is shown.
`...`	Additional arguments passed to `plot()`.

Value

The input object, invisibly.

Examples

```
set.seed(42)
x <- matrix(rnorm(200 * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(200)
x_new <- matrix(rnorm(50 * 3), ncol = 3)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
plot(result)
```

predict.predictset_class
Predict Method for Classification Conformal Objects

Description

Generate prediction sets for new data using a fitted conformal prediction object.

Usage

```
## S3 method for class 'predictset_class'
predict(object, newdata, ...)
```

Arguments

`object`	A `predictset_class` object.
`newdata`	A numeric matrix or data frame of new predictor variables.
`...`	Additional arguments. For Mondrian objects, pass `groups_new` (a factor or character vector of group labels for each observation in `newdata`).

Value

A `predictset_class` object with updated sets and probabilities.

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(ifelse(x[,1] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                 family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)

result <- conformal_lac(x, y, model = clf, x_new = x_new)
preds <- predict(result, newdata = matrix(rnorm(5 * 4), ncol = 4))
```

`predict.predictset_reg` *Predict Method for Regression Conformal Objects*

Description

Generate prediction intervals for new data using a fitted conformal prediction object.

Usage

```
## S3 method for class 'predictset_reg'
predict(object, newdata, ...)
```

Arguments

`object`	A `predictset_reg` object.
`newdata`	A numeric matrix or data frame of new predictor variables.
`...`	Additional arguments. For Mondrian objects, pass `groups_new` (a factor or character vector of group labels for each observation in `newdata`).

Value

A data frame with columns `pred`, `lower`, and `upper`.

Examples

```
set.seed(42)
x <- matrix(rnorm(200 * 3), ncol = 3)
y <- x[, 1] * 2 + rnorm(200)
x_new <- matrix(rnorm(10 * 3), ncol = 3)

result <- conformal_split(x, y, model = y ~ ., x_new = x_new)
preds <- predict(result, newdata = matrix(rnorm(5 * 3), ncol = 3))
```

print.predictset_aci *Print Method for ACI Objects*

Description

Print Method for ACI Objects

Usage

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

Arguments

`x`	A `predictset_aci` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

print.predictset_class *Print Method for Classification Conformal Objects*

Description

Print Method for Classification Conformal Objects

Usage

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

Arguments

`x`	A `predictset_class` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

`print.predictset_model` *Print Method for Model Specifications*

Description

Print Method for Model Specifications

Usage

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

Arguments

`x`	A `predictset_model` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

`print.predictset_reg` *Print Method for Regression Conformal Objects*

Description

Print Method for Regression Conformal Objects

Usage

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

Arguments

`x`	A `predictset_reg` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

`set_size` *Prediction Set Sizes*

Description

Returns the number of classes in each prediction set.

Usage

```
set_size(object)
```

Arguments

`object`	A `predictset_class` object.

Value

An integer vector of set sizes.

See Also

Other diagnostics: `conformal_compare()`, `conformal_pvalue()`, `coverage()`, `coverage_by_bin()`, `coverage_by_group()`, `interval_width()`

Examples

```
set.seed(42)
n <- 300
x <- matrix(rnorm(n * 4), ncol = 4)
y <- factor(ifelse(x[,1] > 0, "A", "B"))
x_new <- matrix(rnorm(50 * 4), ncol = 4)

clf <- make_model(
  train_fun = function(x, y) glm(y ~ ., data = data.frame(y = y, x),
                                 family = "binomial"),
  predict_fun = function(object, x_new) {
    df <- as.data.frame(x_new)
    names(df) <- paste0("X", seq_len(ncol(x_new)))
    p <- predict(object, newdata = df, type = "response")
    cbind(A = 1 - p, B = p)
  },
  type = "classification"
)

result <- conformal_lac(x, y, model = clf, x_new = x_new)
sizes <- set_size(result)
```

```
table(sizes)
```

---

summary.predictset_class *Summary Method for Classification Conformal Objects*

Description

Summary Method for Classification Conformal Objects

Usage

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

Arguments

`object`	A `predictset_class` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

---

summary.predictset_reg *Summary Method for Regression Conformal Objects*

Description

Summary Method for Regression Conformal Objects

Usage

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

Arguments

`object`	A `predictset_reg` object.
`...`	Additional arguments (currently unused).

Value

The input object, invisibly.

Index

∗ **classification methods**
  conformal_aps, 4
  conformal_lac, 14
  conformal_mondrian_class, 17
  conformal_raps, 19
∗ **diagnostics**
  conformal_compare, 7
  conformal_pvalue, 18
  coverage, 24
  coverage_by_bin, 25
  coverage_by_group, 26
  interval_width, 27
  set_size, 35
∗ **regression methods**
  conformal_aci, 3
  conformal_cqr, 9
  conformal_cv, 10
  conformal_jackknife, 12
  conformal_mondrian, 15
  conformal_split, 21
  conformal_weighted, 23

barplot(), *30*

conformal_aci, 3, *10*, *12*, *13*, *16*, *22*, *24*
conformal_aps, 4, *15*, *18*, *20*
conformal_class_split, 6
conformal_compare, 7, *19*, *25–27*, *35*
conformal_cqr, *3*, 9, *12*, *13*, *16*, *22*, *24*
conformal_cv, *3*, *10*, 10, *13*, *16*, *22*, *24*
conformal_jackknife, *3*, *10*, *12*, 12, *16*, *22*, *24*
conformal_lac, *5*, 14, *18*, *20*
conformal_lac(), *5*, *6*, *17*, *20*
conformal_mondrian, *3*, *10*, *12*, *13*, 15, *22*, *24*
conformal_mondrian_class, *5*, *15*, 17, *20*
conformal_pvalue, *8*, 18, *25–27*, *35*
conformal_raps, *5*, *15*, *18*, 19
conformal_split, *3*, *10*, *12*, *13*, *16*, 21, *24*
conformal_split(), *9*, *11*, *13*, *16*, *23*
conformal_weighted, *3*, *10*, *12*, *13*, *16*, *22*, 23
coverage, *8*, *19*, 24, *25–27*, *35*
coverage_by_bin, *8*, *19*, *25*, 25, *26*, *27*, *35*
coverage_by_group, *8*, *19*, *25*, 26, *27*, *35*

interval_width, *8*, *19*, *25*, *26*, 27, *35*

lm(), *11*, *13*, *21*, *22*

make_model, 28
make_model(), *4*, *6*, *8*, *9*, *11*, *13*, *14*, *16*, *17*, *20–23*

plot(), *31*
plot.predictset_aci, 29
plot.predictset_class, 29
plot.predictset_reg, 30
predict(), *11*
predict.predictset_class, 31
predict.predictset_reg, 32
print.predictset_aci, 33
print.predictset_class, 33
print.predictset_model, 34
print.predictset_reg, 34

set_size, *8*, *19*, *25–27*, 35
summary.predictset_class, 36
summary.predictset_reg, 36