Package 'triptych'

May 8, 2026

Title Diagnostic Graphics to Evaluate Forecast Performance

Version 0.1.3

URL https://github.com/aijordan/triptych/,
https://aijordan.github.io/triptych/

Description Overall predictive performance is measured by a mean score (or loss), which decomposes into miscalibration, discrimination, and uncertainty components. The main focus is visualization of these distinct and complementary aspects in joint displays.
See Dimitriadis, Gneiting, Jordan, Vogel (2024) <doi:10.1016/j.ijforecast.2023.09.007>.

License MIT + file LICENSE

Depends R (>= 4.2)

Imports ggplot2, patchwork, pROC, monotone, tidyr, vctrs, dplyr, purrr, tibble, rlang, tidyselect, class, scales, geomtextpath (>= 0.1.4), ggrepel

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.1

SystemRequirements C++20

LinkingTo cpp11

LazyData true

NeedsCompilation yes

Author Timo Dimitriadis [aut, cph], Alexander I. Jordan [aut, cre, cph]

Maintainer Alexander I. Jordan <alexander.jordan@h-its.org>

Repository CRAN

Date/Publication 2024-06-13 15:50:02 UTC

Contents

`accessors` *Accessing original forecast and observation data for triptych objects*

Description

Accessing original forecast and observation data for triptych objects

Usage

```
forecasts(x, ...)

observations(x, ...)
```

Arguments

`x`	An object from which the relevant information should be extracted.
`...`	Additional arguments passed to other methods.

Value

For `forecasts()`: A tibble of the original forecasts in long format.

For `observations()`: A vector of the observations.

See Also

`estimates()`, `regions()`

Examples

```
data(ex_binary, package = "triptych")
tr <- triptych(ex_binary)

forecasts(tr)
observations(tr)
```

`add_confidence` *Adding confidence regions*

Description

Confidence regions are supposed to contain the true "parameter" with a given degree of confidence. Here, "parameter" refers to a murphy curve, a reliability curve, or a ROC curve, respectively.

Usage

```
add_confidence(x, level = 0.9, method = "resampling_cases", ...)
```

Arguments

`x`	An object to which a confidence region should be added.
`level`	A single value for the level of confidence.
`method`	A string that gives the name of method to generate the confidence regions. Currently, one of: "resampling_cases", "resampling_Bernoulli".
`...`	Additional arguments passed to methods.

Value

The object given to `x`, but with information about the confidence regions. This information can be accessed conveniently by using `regions()` on the curve component of interest.

See Also

`resampling_cases()`, `resampling_Bernoulli()`

Examples

```
data(ex_binary, package = "triptych")

tr <- triptych(ex_binary) |>
  dplyr::slice(1, 9)

# Bootstrap resampling is expensive
# (the number of bootstrap samples is small to keep execution times short)

tr <- add_confidence(tr, level = 0.9, method = "resampling_cases", n_boot = 20)
```

```
regions(tr$murphy)
regions(tr$reliability)
regions(tr$roc)
```

`add_consistency` *Adding consistency regions for reliability curves*

Description

Consistency regions are created under the assumption that the forecasts are calibrated. A reliability curve that significantly violates the consistency region indicates a miscalibrated forecast.

Usage

```
add_consistency(x, level = 0.9, method = "resampling_Bernoulli", ...)
```

Arguments

`x`	An object to which a consistency region should be added.
`level`	A single value for the level of confidence.
`method`	A string that gives the name of method to generate the consistency regions. Currently, only: "resampling_Bernoulli".
`...`	Additional arguments passed to methods.

Value

The object given to `x`, but with information about the consistency regions. This information can be accessed conveniently by using `regions()` on the reliability curve component.

See Also

`resampling_Bernoulli()`

Examples

```
data(ex_binary, package = "triptych")

tr <- triptych(ex_binary) |>
  dplyr::slice(1, 9)

# Bootstrap resampling is expensive
# (the number of bootstrap samples is small to keep execution times short)

tr <- add_consistency(tr, level = 0.9, method = "resampling_Bernoulli", n_boot = 20)
regions(tr$reliability)
```

`estimates` *Accessing diagnostic estimate data*

Description

Accessing diagnostic estimate data

Usage

```
estimates(x, at, ...)

## S3 method for class 'triptych_mcbdsc'
estimates(x, ...)

## S3 method for class 'triptych_murphy'
estimates(x, at = NULL, ...)

## S3 method for class 'triptych_reliability'
estimates(x, at = NULL, ...)

## S3 method for class 'triptych_roc'
estimates(x, at = NULL, p1 = mean(observations(x)), ...)
```

Arguments

`x`	An object from which the estimate information should be extracted.
`at`	A vector of thresholds where `x` should be evaluated.
`...`	Additional arguments passed to other methods.
`p1`	The unconditional event probability. Used in combination with `at` to determine the diagonal lines along which to determine the estimate.

Value

A tibble with the relevant information describing the diagnostic estimate (Murphy curve, reliability curve, ROC curve, score decomposition) for all supplied forecasting methods.

For a Murphy curve, a tibble with columns: `forecast`, `knot` (the threshold value), `limit` ("left" or "right" in `knot`, only present if `at = NULL`), `mean_score`.

For a reliability curve, a tibble with columns: `forecast`, `CEP`, `x` (the knots of the isotonic regression estimate).

For a ROC curve, a tibble with columns: `forecast`, `FAR` (false alarm rate), `HR` (hit rate).

For a MCBDSC decomposition, a tibble with columns: `forecast`, `mean_score`, `MCB` (miscalibration), `DSC` (discrimination), `UNC` (uncertainty).

See Also

`regions()`, `forecasts()`, `observations()`

Examples

```
data(ex_binary, package = "triptych")
tr <- triptych(ex_binary)

estimates(tr$murphy)
estimates(tr$reliability)
estimates(tr$roc)
estimates(tr$mcbdsc)
```

`ex_binary` *Example data set of binary observations and probability forecasts*

Description

The forecasts X01 to X10 are generated in such a way that their discrimination ability is neatly decreasing. In addition, X01 and X06 are "calibrated", X02 and X07 are "underconfident", X03 and X08 are "overconfident", X04 and X09 exhibit "negative bias", and X05 and X10 exhibit "positive bias".

Usage

```
ex_binary
```

Format

A data frame with 1,000 rows and 11 columns, generated as described in 'Details':

y observations

X01 forecasts, full information, calibrated: $a = 1$, $b = 1$

X02 forecasts, less information than X01, underconfident: $a = 1/4$, $b = 1/4$

X03 forecasts, less information than X02, overconfident: $a = 4$, $b = 4$

X04 forecasts, less information than X03, negative bias: $a = 5/3$, $b = 3/5$

X05 forecasts, less information than X04, positive bias: $a = 3/5$, $b = 5/3$

X06 forecasts, less information than X05, calibrated: $a = 1$, $b = 1$

X07 forecasts, less information than X06, underconfident: $a = 1/4$, $b = 1/4$

X08 forecasts, less information than X07, overconfident: $a = 4$, $b = 4$

X09 forecasts, less information than X08, negative bias: $a = 5/3$, $b = 3/5$

X10 forecasts, least information, positive bias: $a = 2/3$, $b = 3/2$

Details

The observations are generated from a Bernoulli distribution, where the success probability is determined by ten sources of information. That is, the probability is given by

$$p = \Phi(\sum_{i=1}^{10} Z_i),$$

where Z_i, $i = 1, ..., 10$, are independent standard Gaussian random variables, and Φ denotes the cumulative distribution function of the standard Gaussian distribution.

The corresponding forecasts are named in decreasing order of access to these latent Gaussian variables (that is, information content). In a first step, calibrated forecasts are generated by $p[j] = \Phi(\frac{1}{j}\sum_{i=j}^{10} Z_i)$. Subsequently, these probabilities are perturbed to introduce miscalibration using the cumulative distribution function F of the beta distribution, yielding the final forecasts

$$X[j] = F(p[j]; a, b),$$

where a and b are the positive shape parameters (see `pbeta()`).

`mcbdsc` *Evaluation of forecasts using score decompositions*

Description

A score decomposition splits the mean score into the three components of miscalibration (MCB), discrimination (DSC), and uncertainty (UNC). Plotting the DSC component against the MCB component allows for a quick visual inspection of predictive performance for many forecasting methods.

Usage

```
mcbdsc(x, y_var = "y", ..., y = NULL, score = "Brier_score")

as_mcbdsc(x, r)
```

Arguments

`x`	A data frame, list, matrix, or other object that can be coerced to a tibble. Contains numeric forecasts, and observations (optional).
`y_var`	A variable in `x` that contains observations. Specified as the argument `var` in `dplyr::pull()`.
`...`	Unused.
`y`	A numeric vector of observations. If supplied, overrides `y_var`. Otherwise, defaults to `dplyr::pull(x, y_var)`.
`score`	A string specifying the score function. One of: `"Brier_score"` (default), `"log_score"`, `"MR_score"`.
`r`	A reference triptych_mcbdsc object whose attributes are used for casting.

Value

A `triptych_mcbdsc` object, that is a `vctrs_vctr` subclass, and has a length equal to number of forecasting methods supplied in `x`. Each entry is named according to the corresponding forecasting method, and contains a list of named objects:

- `estimate`: A data frame of the score decomposition.
- `region`: An empty list. Adding confidence regions is not yet supported.
- `x`: The numeric vector of original forecasts.

Access is most convenient through `estimates()`, `regions()`, and `forecasts()`.

See Also

Accessors: `estimates()`, `regions()`, `forecasts()`, `observations()`

Visualization: `plot.triptych_mcbdsc()`, `autoplot.triptych_mcbdsc()`

Examples

```
data(ex_binary, package = "triptych")

md <- mcbdsc(ex_binary)
md

autoplot(md)
estimates(md)
```

`murphy` *Evaluation of forecasts using Murphy curves*

Description

A Murphy curve visualizes economic utility by displaying the mean elementary scores across all threshold values.

Usage

```
murphy(x, y_var = "y", ref_var = "ref", ..., y = NULL, ref = NULL)

as_murphy(x, r)
```

Arguments

x	A data frame, list, matrix, or other object that can be coerced to a tibble. Contains numeric forecasts, and observations (optional).
y_var	A variable in x that contains observations. Specified as the argument varin dplyr::pull().
ref_var	A variable in x that contains reference forecasts. Specified as the argument var in dplyr::pull()). Ignored, if the default name is not present in x.
...	Unused.
y	A numeric vector of observations. If supplied, overrides y_var. Otherwise, defaults to dplyr::pull(x, y_var).
ref	A numeric vector of reference forecasts. If supplied, overrides ref_var. Otherwise, ignored (see ref_var) or dplyr::pull(x, ref_var).
r	A reference triptych_murphy object whose attributes are used for casting.

Value

A triptych_murphy object, that is a vctrs_vctr subclass, and has a length equal to number of forecasting methods supplied in x. Each entry is named according to the corresponding forecasting method, and contains a list of named objects:

- estimate: A data frame with the threshold and corresponding mean score values.
- region: Either an empty list, or a data frame of point confidence intervals added by add_confidence().
- x: The numeric vector of original forecasts.

Access is most convenient through estimates(), regions(), and forecasts().

See Also

Accessors: estimates(), regions(), forecasts(), observations()

Adding uncertainty quantification: add_confidence()

Visualization: plot.triptych_murphy(), autoplot.triptych_murphy()

Examples

```
data(ex_binary, package = "triptych")

mr <- murphy(ex_binary)
mr

# 1. Choose 4 predictions
# 2. Visualize
# 3. Adjust the title of the legend
mr[c(1, 3, 6, 9)] |>
  autoplot() +
  ggplot2::guides(colour = ggplot2::guide_legend("Forecast"))

# Build yourself using accessors
```

```
library(ggplot2)
df_est <- estimates(mr[c(1, 3, 6, 9)])
ggplot(df_est) +
  geom_path(aes(x = knot, y = mean_score, col = forecast))
```

`plot.triptych` *Plot methods for the triptych classes*

Description

Plot methods for the triptych classes

Usage

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

## S3 method for class 'triptych'
autoplot(object, ...)

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

## S3 method for class 'triptych_murphy'
autoplot(object, ...)

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

## S3 method for class 'triptych_reliability'
autoplot(object, ..., breaks = seq(0, 1, length.out = 11))

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

## S3 method for class 'triptych_roc'
autoplot(object, ...)

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

## S3 method for class 'triptych_mcbdsc'
autoplot(
  object,
  ...,
  n_isolines = 10,
```

```
  colour_values = "black",
  colour_unc = "#00BF7D",
  MCBDSC_repel = FALSE,
  MCB_lim = NA,
  DSC_lim = NA
)
```

Arguments

x	An object that inherits from one of the triptych classes.
...	Arguments passed from `autoplot.triptych()` to the other methods for triptych classes.
object	An object that inherits from one of the triptych classes.
breaks	A vector of bin boundaries for the `geom_histogram()` layer. Set to `NA` to disable.
n_isolines	The number of isolines showing mean scores.
colour_values	A colour specification passed to the `values` argument of `scale_colour_manual()`. Recycled if length 1.
colour_unc	A colour specification highlighting the UNC component layers.
MCBDSC_repel	A boolean value indicating whether labels should be placed by the `ggrepel` package.
MCB_lim	The plot limits for the x-axis (the MCB component).
DSC_lim	The plot limits for the y-axis (the DSC component).

Value

For an object of class `'triptych'`: A patchwork object (invisibly).

For all other triptych objects: A ggplot object (invisibly).

Every `plot()` method wraps the corresponding `autoplot()` method, followed by an explicit `print()` call. That is, it always draws a plot, even during assignment or within a loop.

Examples

```
data(ex_binary, package = "triptych")
tr <- triptych(ex_binary)

dplyr::slice(tr, 1, 3, 6, 9) |> autoplot()
autoplot(tr$murphy)
autoplot(tr$reliability)
autoplot(tr$roc)
autoplot(tr$mcbdsc)
```

regions *Accessing confidence/consistency region data*

Description

Accessing confidence/consistency region data

Usage

```
regions(x, ...)

## S3 method for class 'triptych_mcbdsc'
regions(x, ...)

## S3 method for class 'triptych_murphy'
regions(x, ...)

## S3 method for class 'triptych_reliability'
regions(x, ...)

## S3 method for class 'triptych_roc'
regions(x, ...)
```

Arguments

x	An object from which the region information should be extracted.
...	Additional arguments passed to other methods.

Value

A tibble with the relevant information for the uncertainty quantification of the chosen diagnostic (Murphy curve, reliability curve, ROC curve, score decomposition) for all supplied forecasting methods.

For a Murphy curve, a tibble with columns: `forecast`, `threshold`, `lower`, `upper`, `method`, `level`.

For a reliability curve, a tibble with columns: `forecast`, `x` (forecast values), `lower`, `upper`, `method`, `level`.

For a ROC curve, a tibble with columns: `forecast`, `FAR` (false alarm rate), `HR` (hit rate), `method`, `level`. This tibble is twice as long as those for Murphy and reliability curves, since the FAR-HR pairs are ordered to describe a polygon, generated by pointwise confidence intervals along diagonal lines with slope $-\pi_0/\pi_1$. Here, $\pi_1 = 1 - \pi_0$ is the unconditional event probability.

See Also

`estimates()`, `forecasts()`, `observations()`

Examples

```
data(ex_binary, package = "triptych")

# Bootstrap resampling is expensive
# (the number of bootstrap samples is small to keep execution times short)

tr <- triptych(ex_binary) |>
  dplyr::slice(1, 9) |>
  add_confidence(level = 0.9, method = "resampling_cases", n_boot = 20)

regions(tr$murphy)
regions(tr$reliability)
regions(tr$roc)
```

`reliability` *Evaluation of forecasts using reliability curves*

Description

A reliability curve visualizes miscalibration by displaying the (isotonic) conditional event probability against the forecast value.

Usage

```
reliability(x, y_var = "y", ..., y = NULL)

as_reliability(x, r)
```

Arguments

`x`	A data frame, list, matrix, or other object that can be coerced to a tibble. Contains numeric forecasts, and observations (optional).
`y_var`	A variable in `x` that contains observations. Specified as the argument `var`in `dplyr::pull()`.
`...`	Unused.
`y`	A numeric vector of observations. If supplied, overrides `y_var`. Otherwise, defaults to `dplyr::pull(x, y_var)`.
`r`	A reference triptych_mcbdsc object whose attributes are used for casting.

Value

A `triptych_reliability` object, that is a `vctrs_vctr` subclass, and has a length equal to number of forecasting methods supplied in `x`. Each entry is named according to the corresponding forecasting method, and contains a list of named objects:

- `estimate`: A data frame with the isotonic regression estimate.

- `region`: Either an empty list, or a data frame of pointwise consistency or confidence intervals added by `add_consistency()` or `add_confidence()`, respectively.
- `x`: The numeric vector of original forecasts.

Access is most convenient through `estimates()`, `regions()`, and `forecasts()`.

See Also

Accessors: `estimates()`, `regions()`, `forecasts()`, `observations()`

Adding uncertainty quantification: `add_confidence()`

Visualization: `plot.triptych_reliability()`, `autoplot.triptych_reliability()`

Examples

```
data(ex_binary, package = "triptych")

rel <- reliability(ex_binary)
rel

# 1. Choose 4 predictions
# 2. Visualize
# 3. Adjust the title of the legend
rel[c(1, 3, 6, 9)] |>
  autoplot() +
  ggplot2::guides(colour = ggplot2::guide_legend("Forecast"))

# Build yourself using accessors
library(ggplot2)
df_est <- estimates(rel[c(1, 3, 6, 9)])
ggplot(df_est, aes(x = x, y = CEP, col = forecast)) +
  geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1)) +
  geom_path()
```

`resampling_Bernoulli` *Bootstrap (binary) observation resampling for triptych objects*

Description

This function is intended to be called from `add_consistency()` or `add_confidence()`, by specifying `"resampling_Bernoulli"` in the respective `method` argument.

Usage

```
resampling_Bernoulli(x, level = 0.9, n_boot = 1000, ...)

## S3 method for class 'triptych_murphy'
resampling_Bernoulli(x, level = 0.9, n_boot = 1000, ...)
```

```
## S3 method for class 'triptych_reliability'
resampling_Bernoulli(
  x,
  level = 0.9,
  n_boot = 1000,
  position = c("diagonal", "estimate"),
  ...
)

## S3 method for class 'triptych_roc'
resampling_Bernoulli(x, level = 0.9, n_boot = 1000, ...)
```

Arguments

x	One of the triptych objects.
level	A single value that determines which quantiles of the bootstrap sample to return. These quantiles envelop level * n_boot bootstrap draws.
n_boot	The number of bootstrap samples.
...	Additional arguments passed to other methods.
position	Either "estimate" for confidence regions, or "diagonal" for consistency regions.

Details

Bootstrap (binary) observation resampling assumes conditionally independent observations given the forecast value. A given number of bootstrap samples are the basis for pointwise computed confidence/consistency intervals. For every bootstrap sample, we sample observations from a Bernoulli distribution conditional on (recalibrated) forecast values.

Value

A list of tibbles that contain the information to draw confidence regions. The length is equal to the number of forecasting methods in x.

Examples

```
data(ex_binary, package = "triptych")

# Bootstrap resampling is expensive
# (the number of bootstrap samples is small to keep execution times short)

tr_consistency <- triptych(ex_binary) |>
  dplyr::slice(1, 9) |>
  add_consistency(level = 0.9, method = "resampling_Bernoulli", n_boot = 20)

tr_confidence <- triptych(ex_binary) |>
  dplyr::slice(1, 9) |>
  add_confidence(level = 0.9, method = "resampling_Bernoulli", n_boot = 20)
```

resampling_cases *Bootstrap case resampling for triptych objects*

Description

This function is intended to be called from add_confidence(), by specifying "resampling_cases" in the method argument.

Usage

```
resampling_cases(x, level = 0.9, n_boot = 1000, ...)

## S3 method for class 'triptych_murphy'
resampling_cases(x, level = 0.9, n_boot = 1000, ...)

## S3 method for class 'triptych_reliability'
resampling_cases(x, level = 0.9, n_boot = 1000, ...)

## S3 method for class 'triptych_roc'
resampling_cases(x, level = 0.9, n_boot = 1000, ...)
```

Arguments

x	One of the triptych objects.
level	A single value that determines which quantiles of the bootstrap sample to return. These quantiles envelop level * n_boot bootstrap draws.
n_boot	The number of bootstrap samples.
...	Additional arguments passed to other methods.

Details

Case resampling assumes independent and identically distributed forecast-observation pairs. A given number of bootstrap samples are the basis for pointwise computed confidence intervals. For every bootstrap sample, we draw forecast-observations pairs with replacement until the size of the original data set is reached.

Value

A list of tibbles that contain the information to draw confidence regions. The length is equal to the number of forecasting methods in x.

Examples

```
data(ex_binary, package = "triptych")

# Bootstrap resampling is expensive
# (the number of bootstrap samples is small to keep execution times short)
```

```
tr <- triptych(ex_binary) |>
  dplyr::slice(1, 9) |>
  add_confidence(level = 0.9, method = "resampling_cases", n_boot = 20)
```

---

roc *Evaluation of forecasts using ROC curves*

Description

A ROC curve visualizes discrimination ability by displaying the hit rate against the false alarm rate for all threshold values.

Usage

```
roc(x, y_var = "y", ..., y = NULL, concave = TRUE)

as_roc(x, r)
```

Arguments

x	A data frame, list, matrix, or other object that can be coerced to a tibble. Contains numeric forecasts, and observations (optional).
y_var	A variable in x that contains observations. Specified as the argument varin dplyr::pull().
...	Unused.
y	A numeric vector of observations. If supplied, overrides y_var. Otherwise, defaults to dplyr::pull(x, y_var).
concave	A boolean value indicating whether to calculate the concave hull or the raw ROC diagnostic.
r	A reference triptych_mcbdsc object whose attributes are used for casting.

Value

A triptych_roc object, that is a vctrs_vctr subclass, and has a length equal to number of forecasting methods supplied in x. Each entry is named according to the corresponding forecasting method, and contains a list of named objects:

- estimate: A data frame of hit rates and false rates.
- region: Either an empty list, or a data frame of pointwise confidence intervals (along diagonal lines with slope $-\pi_0/\pi_1$) added by add_confidence().
- x: The numeric vector of original forecasts.

Access is most convenient through estimates(), regions(), and forecasts().

See Also

Accessors: `estimates()`, `regions()`, `forecasts()`, `observations()`

Adding uncertainty quantification: `add_confidence()`

Visualization: `plot.triptych_roc()`, `autoplot.triptych_roc()`

Examples

```
data(ex_binary, package = "triptych")

rc <- roc(ex_binary)
rc

# 1. Choose 4 predictions
# 2. Visualize
# 3. Adjust the title of the legend
rc[c(1, 3, 6, 9)] |>
  autoplot() +
  ggplot2::guides(colour = ggplot2::guide_legend("Forecast"))

# Build yourself using accessors
library(ggplot2)
df_est <- estimates(rc[c(1, 3, 6, 9)])
ggplot(df_est, aes(x = FAR, y = HR, col = forecast)) +
  geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1)) +
  geom_path()
```

`triptych` *Evaluation of forecasts using a Triptych*

Description

A triptych visualizes three important aspects of predictive performance: Economic utility via Murphy curves, miscalibration via reliability curves, and discrimination ability via ROC curves. The `triptych` S3 class has plotting methods for `ggplot2`.

Usage

```
triptych(x, y_var = "y", ..., y = NULL)
```

Arguments

`x`	A data frame, list, matrix, or other object that can be coerced to a tibble. Contains numeric forecasts, and observations (optional).
`y_var`	A variable in `x` that contains observations. Specified as the argument `var` in `dplyr::pull()`.
`...`	Additional arguments passed to `murphy()`, `reliability()`, `roc()`, and `mcbdsc()`.
`y`	A numeric vector of observations. If supplied, overrides `y_var`. Otherwise, defaults to `dplyr::pull(x, y_var)`.

Value

A `triptych` object, that is a tibble subclass, and contains five columns:

- `forecast`: Contains the names.
- `murphy`: Contains a `vctrs_vctr` subclass of Murphy curves.
- `reliability`: Contains a `vctrs_vctr` subclass of reliability curves.
- `roc`: Contains a `vctrs_vctr` subclass of ROC curves.
- `mcbdsc`: Contains a `vctrs_vctr` subclass of score decompositions.

See Also

Vector class constructors: `murphy()`, `reliability()`, `roc()`, `mcbdsc()`

Adding uncertainty quantification: `add_consistency()`, `add_confidence()`

Visualization: `plot.triptych()`, `autoplot.triptych()`

Examples

```
data(ex_binary, package = "triptych")

tr <- triptych(ex_binary)
identical(tr, triptych(ex_binary, y))
identical(tr, triptych(ex_binary, 1))
tr

# 1. Choose 4 predictions
# 2. Add consistency bands (for reliability curves)
#    (Bootstrap resampling is expensive, the number of bootstrap samples
#     is small to keep execution times short)
# 3. Create patchwork object
# 4. Adjust the title of the legend
dplyr::slice(tr, 1, 3, 6, 9) |>
  add_consistency(level = 0.9, method = "resampling_Bernoulli", n_boot = 20) |>
  autoplot() &
  ggplot2::guides(colour = ggplot2::guide_legend("Forecast"))
```

Index