Package ‘generics’

October 25, 2021

Title Common S3 Generics not Provided by Base R Methods Related to Model Fitting

Version 0.1.1

Description In order to reduce potential package dependencies and conflicts, generics provides a number of commonly used S3 generics.

License MIT + file LICENSE

URL `https://generics.r-lib.org`, `https://github.com/r-lib/generics`

BugReports `https://github.com/r-lib/generics/issues`

Depends R (¿= 3.2)

Imports methods

Suggests covr,
pkgload,
testthat (¿= 3.0.0),
tibble,
withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

R topics documented:

`augment` *Augment data with information from an object*

Description

Augment data with information from an object

Usage

```
augment(x, ...)
```

Arguments

`x`	Model object or other R object with information to append to observations.
`...`	Addition arguments to `augment` method.

Value

A `tibble::tibble()` with information about data points.

Methods

No methods found in currently loaded packages.

`calculate` *Calculate statistics.*

Description

Calculate statistics.

Usage

```
calculate(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`coercion-factor` *Factor coercion*

Description

Coercion functions for creating factors from other existing objects.

Usage

```
as.factor(x, ...)

as.ordered(x, ...)
```

Arguments

`x`	A vector of data.
`...`	Other arguments passed on to methods.

Details

These functions override non-generic factor coercion functions provided in base so that packages can provide methods for different data types. The default methods call the base versions.

Value

For `as.factor()`, a factor. For `as.ordered()`, an ordered factor.

Methods

`as.factor()`: No methods found in currently loaded packages.

`as.ordered()`: No methods found in currently loaded packages.

Examples

```
as.factor(letters[1:5])
as.ordered(letters[1:5])
```

`coercion-time-difference`

Time difference coercion

Description

Coercion functions for creating `difftime` objects from other existing objects.

Usage

```
as.difftime(tim, ...)

## Default S3 method:
as.difftime(tim, format = "%X", units = "auto", ...)
```

Arguments

`tim`	A vector specifying a time interval.
`...`	Other arguments passed on to methods.
`format`	A single character specifying the format of `tim` when it is a character. The default is a locale-specific time format.
`units`	A single character specifying units in which the results are desired. Required if `tim` is a numeric.

Details

This function overrides the non-generic `as.difftime()` function provided in base so that packages can provide methods for different data types. The default method call the base version.

Value

A `difftime` object with an attribute indicating the units.

Methods

See the following help topics for more details about individual methods:

`generics`

- `coercion-time-difference`: `default`

Examples

```
as.difftime(1:5, units = "secs")

as.difftime(c("01:55:22", "01:55:25"))

as.difftime("01", format = "%H")
as.difftime("01", format = "%H", units = "secs")
```

`compile` *Configure an object*

Description

Finalizes or completes an object.

Usage

```
compile(object, ...)
```

Arguments

`object`	An object. See the individual method for specifics.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`components` *Extract components*

Description

`components` can be used to extract elements from an object.

Usage

```
components(object, ...)
```

Arguments

`object`	A data separable object.
`...`	Other arguments passed to methods

Details

For example, decomposition methods and some modelling techniques can be used to decompose a dataset into components of interest. This function is used to extract these components in a tidy data format.

Value

A dataset (`tibble::tibble()` or similar) containing components from the object.

Methods

No methods found in currently loaded packages.

`equation` *Model equations*

Description

Display the mathematical representation of a fitted model.

Usage

```
equation(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Other arguments passed to methods

Value

Markup output suitable for rendering the equation.

Methods

No methods found in currently loaded packages.

`estfun` *Extracting the estimating functions of a fitted model.*

Description

Extracting the estimating functions of a fitted model.

Usage

```
estfun(x, ...)
```

Arguments

`x`	A fitted model object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`evaluate` *Evaluate an object.*

Description

Evaluate an object.

Usage

```
evaluate(x, ...)
```

Arguments

`x`	An object. See the individual method for specifics.
`...`	other arguments passed to methods

Methods

No methods found in currently loaded packages.

`explain` *Explain details of an object*

Description

Explain details of an object

Usage

```
explain(x, ...)
```

Arguments

`x`	An object. See the individual method for specifics.
`...`	other arguments passed to methods

Methods

No methods found in currently loaded packages.

explore — *Create an interactive visualization appropriate to a particular object type*

Description

`explore()` invokes a function that starts an interactive, pre-defined widget (e.g. `plotly` visualization, `shiny` app, etc.) to investigate the results.

Usage

```
explore(x, ...)
```

Arguments

`x`	A object
`...`	Other arguments passed to methods

Value

`NULL` (invisibly) or some other data type (e.g. tibble) depending on the application.

Methods

No methods found in currently loaded packages.

fit — *Estimate model parameters.*

Description

Estimates parameters for a given model from a set of data.

Usage

```
fit(object, ...)
```

Arguments

`object`	An object. See the individual method for specifics.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`fit_xy` *Estimate model parameters.*

Description

Estimates parameters for a given model from a set of data in the form of a set of predictors (`x`) and outcome(s) (`y`).

Usage

```
fit_xy(object, ...)
```

Arguments

`object`	An object. See the individual method for specifics.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`generate` *Generate values based on inputs*

Description

Generate values based on inputs

Usage

```
generate(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`glance` *Glance at an object*

Description

Construct a single row summary "glance" of a model, fit, or other object

Usage

```
glance(x, ...)
```

Arguments

`x`	model or other R object to convert to single-row data frame
`...`	other arguments passed to methods

Details

glance methods always return either a one-row data frame (except on `NULL`, which returns an empty data frame)

Methods

No methods found in currently loaded packages.

`hypothesize` *Construct hypotheses.*

Description

Construct hypotheses.

Usage

```
hypothesize(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`interpolate` *Interpolate missing values*

Description

Interpolates missing values provided in the training dataset using the fitted model.

Usage

```
interpolate(object, ...)
```

Arguments

`object`	A fitted model object
`...`	Other arguments passed to methods

Value

A dataset (`tibble::tibble()` or similar) of the same structure as the input dataset with missing values from the response variable replaced with interpolated values.

Methods

No methods found in currently loaded packages.

`learn` *Estimate model parameters.*

Description

Estimates parameters for a given model from a set of data.

Usage

```
learn(x, ...)
```

Arguments

`x`	An object. See the individual method for specifics.
`...`	other arguments passed to methods

Methods

No methods found in currently loaded packages.

min_grid — *Determine the minimum set of model fits*

Description

`min_grid()` determines exactly what models should be fit in order to evaluate the entire set of tuning parameter combinations. This is for internal use only and the API may change in the near future.

Usage

```
min_grid(x, grid, ...)
```

Arguments

`x`	A model specification.
`grid`	A tibble with tuning parameter combinations.
`...`	Not currently used.

Value

A tibble with the minimum tuning parameters to fit and an additional list column with the parameter combinations used for prediction.

Methods

No methods found in currently loaded packages.

prune — *Prune or reduce an object*

Description

Prune or reduce an object

Usage

```
prune(tree, ...)
```

Arguments

`tree`	A fitted model object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`refit` *Refitting models*

Description

Refitting models

Usage

```
refit(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`required_pkgs` *Determine packages required by objects*

Description

Determine packages required by objects

Usage

```
required_pkgs(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods

Value

A character string of packages that are required.

Methods

No methods found in currently loaded packages.

`setops` *Set operations*

Description

Union (`union()`), intersect (`intersect()`), difference (`setdiff()`), and equality (`setequal()`) for two vectors representing sets. Determine membership with `is.element()`.

Usage

```
intersect(x, y, ...)

union(x, y, ...)

setdiff(x, y, ...)

setequal(x, y, ...)

is.element(el, set, ...)
```

Arguments

`x, y`	Vectors to combine.
`...`	Other arguments passed on to methods.
`el, set`	Element and set to compare.

Details

These functions override the set functions provided in base to make them generic so that packages can provide methods for different data types. The default methods call the base versions.

Value

For `union()`, `intersect()`, and `setdiff()`, a vector with all duplicate removed.

For `setequal()` and `is.element()`, a logical `TRUE` or `FALSE`.‘

Methods

`intersect()`: No methods found in currently loaded packages.

`union()`: No methods found in currently loaded packages.

`setdiff()`: No methods found in currently loaded packages.

`setequal()`: No methods found in currently loaded packages.

`is.element()`: No methods found in currently loaded packages.

Examples

```
intersect(1:5, 4:8)
union(1:5, 4:8)

setdiff(1:5, 4:8)
setdiff(4:8, 1:5)
```

`specify` *Specify variables or other quantities.*

Description

Specify variables or other quantities.

Usage

```
specify(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`tidy` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

```
tidy(x, ...)
```

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`train` *Estimate model parameters.*

Description

Estimates parameters for a given model from a set of data.

Usage

```
train(x, ...)
```

Arguments

`x`	An object. See the individual method for specifics.
`...`	other arguments passed to methods

Methods

No methods found in currently loaded packages.

`tunable` *Declare tunable parameters*

Description

Returns information on potential hyper-parameters that can be optimized.

Usage

```
tunable(x, ...)
```

Arguments

`x`	An object, such as a recipe, recipe step, workflow, or model specification.
`...`	Other arguments passed to methods

Details

For a model specification, an engine must be chosen.

If the object has no tunable parameters, a tibble with no rows is returned.

The information about the default parameter object takes the form of a named list with an element for the function call and an optional element for the source of the function (e.g. the `dials` package). For model specifications, If the parameter is unknown to the underlying `tunable` method, a `NULL` is returned.

Value

A tibble with a column for the parameter `name`, information on the *default* method for generating a corresponding parameter object, the `source` of the parameter (e.g. "recipe", etc.), and the `component` within the source. For the `component` column, a little more specificity is given about the location of the parameter (e.g. "step_normalize" or recipes or "boost_tree" for models). The `component_id` column contains the unique step `id` field or, for models, a logical for whether the model specification argument was a main parameter or one associated with the engine.

Methods

No methods found in currently loaded packages.

`tune_args` *Determine arguments tagged for tuning*

Description

`tune_args()` takes an object such as a model specification or a recipe and returns a tibble of information on all possible tunable arguments and whether or not they are actually tunable.

Usage

```
tune_args(object, ...)
```

Arguments

`object`	A `model_spec`, `recipe`, `workflow`, or other object.
`...`	Other arguments passed to methods.

Details

The `source` column is determined differently for a `model_spec` or a `recipe` (with additional detail on the type).

The `id` field has any identifier that was passed from `tune::tune()` (e.g. `tune("some note")`). If no additional detail was used in that function, the `id` field reverts to the name of the parameters.

Value

A tibble with columns for the parameter name (`name`), whether it contains *any* tunable value (`tune`), the `id` for the parameter (`id`), and the information on where the parameter was located (`source`).

Methods

No methods found in currently loaded packages.

`varying_args` *Find any arguments that are not fully specified.*

Description

Find any arguments that are not fully specified.

Usage

```
varying_args(object, ...)
```

Arguments

`object`	An object. See the individual method for specifics.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`var_imp` *Calculation of variable importance*

Description

A generic method for calculating variable importance for model objects.

Usage

```
var_imp(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.

`visualize` *Visualize a data set or object.*

Description

Visualize a data set or object.

Usage

```
visualize(x, ...)
```

Arguments

`x`	A data frame or other object.
`...`	Other arguments passed to methods

Methods

No methods found in currently loaded packages.