Package ‘tidyclust’

December 20, 2022

Title A Common API to Clustering

Version 0.1.1

Description A common interface to specifying clustering models, in the same style as 'parsnip'. Creates unified interface across different functions and computational engines.

License MIT + file LICENSE

URL `https://github.com/tidymodels/tidyclust`

BugReports `https://github.com/tidymodels/tidyclust/issues`

Imports cli (>= 3.0.0), dials (>= 1.1.0), dplyr (>= 1.0.9), flexclust (>= 1.3-6), foreach, generics (>= 0.1.2), glue (>= 1.6.2), hardhat (>= 1.0.0), modelenv (>= 0.1.0), parsnip (>= 1.0.2), prettyunits (>= 1.1.0), Rfast (>= 2.0.6), rlang (>= 1.0.6), rsample (>= 1.0.0), stats, tibble (>= 3.1.0), tidyr (>= 1.2.0), tune (>= 1.0.0), utils, vctrs (>= 0.5.0)

Suggests cluster, ClusterR, covr, knitr, modeldata (>= 1.0.0), RcppHungarian, recipes (>= 1.0.0), rmarkdown, testthat (>= 3.0.0), workflows (>= 1.1.2)

Config/Needs/website pkgdown, tidymodels, tidyverse, palmerpenguins, patchwork, ggforce, tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.1.9000

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre] (<`https://orcid.org/0000-0002-0679-1945`>), Kelly Bodwin [aut], Posit Software, PBC. [cph, fnd]

Maintainer Emil Hvitfeldt <`emilhhvitfeldt@gmail.com`>

Repository CRAN

Date/Publication 2022-12-20 03:50:02 UTC

R topics documented:

`augment.cluster_fit` *Augment data with predictions*

Description

`augment()` will add column(s) for predictions to the given data.

Usage

```
## S3 method for class 'cluster_fit'
augment(x, new_data, ...)
```

Arguments

`x`	A `cluster_fit` object produced by `fit.cluster_spec()` or `fit_xy.cluster_spec()` .
`new_data`	A data frame or matrix.
`...`	Not currently used.

Details

For partition models, a `.pred_cluster` column is added.

Value

A `tibble::tibble()` with containing `new_data` with columns added depending on the mode of the model.

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

kmeans_fit %>%
  augment(new_data = mtcars)
```

`cluster_metric_set` *Combine metric functions*

Description

`cluster_metric_set()` allows you to combine multiple metric functions together into a new function that calculates all of them at once.

Usage

```
cluster_metric_set(...)
```

Arguments

`...`	The bare names of the functions to be included in the metric set. These functions must be cluster metrics such as `sse_total()`, `sse_ratio()`, or `silhouette_avg()`.

Details

All functions must be:

- Only cluster metrics

Value

A `cluster_metric_set()` object, combining the use of all input metrics.

`control_cluster` *Control the fit function*

Description

Options can be passed to the `fit.cluster_spec()` function that control the output and computations.

Usage

```
control_cluster(verbosity = 1L, catch = FALSE)
```

Arguments

`verbosity`	An integer where a value of zero indicates that no messages or output should be shown when packages are loaded or when the model is fit. A value of 1 means that package loading is quiet but model fits can produce output to the screen (depending on if they contain their own `verbose`-type argument). A value of 2 or more indicates that any output should be seen.
`catch`	A logical where a value of `TRUE` will evaluate the model inside of `try(, silent = TRUE)`. If the model fails, an object is still returned (without an error) that inherits the class "try-error".

Value

An S3 object with class "control_cluster" that is a named list with the results of the function call

Examples

```
control_cluster()

control_cluster(catch = TRUE)
```

`extract-tidyclust` *Extract elements of a tidyclust model object*

Description

These functions extract various elements from a clustering object. If they do not exist yet, an error is thrown.

- `extract_fit_engine()` returns the engine specific fit embedded within a tidyclust model fit. For example, when using `k_means()` with the `"lm"` engine, this returns the underlying `kmeans` object.
- `extract_parameter_set_dials()` returns a set of dials parameter objects.

Usage

```
## S3 method for class 'cluster_fit'
extract_fit_engine(x, ...)

## S3 method for class 'cluster_spec'
extract_parameter_set_dials(x, ...)
```

Arguments

`x`	A `cluster_fit` object or a `cluster_spec` object.
`...`	Not currently used.

Details

Extracting the underlying engine fit can be helpful for describing the model (via `print()`, `summary()`, `plot()`, etc.) or for variable importance/explainers.

However, users should not invoke the `predict()` method on an extracted model. There may be preprocessing operations that `tidyclust` has executed on the data prior to giving it to the model. Bypassing these can lead to errors or silently generating incorrect predictions.

Good:

```
tidyclust_fit %>% predict(new_data)
```

Bad:

```
tidyclust_fit %>% extract_fit_engine() %>% predict(new_data)
```

Value

The extracted value from the tidyclust object, `x`, as described in the description section.

Examples

```
kmeans_spec <- k_means(num_clusters = 2)
kmeans_fit <- fit(kmeans_spec, ~ ., data = mtcars)

extract_fit_engine(kmeans_fit)
```

`extract_centroids` *Extract clusters from model*

Description

Extract clusters from model

Usage

```
extract_centroids(object, ...)
```

Arguments

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

Value

A `tibble::tibble()` with 1 row for each centroid and their position.

Examples

```
set.seed(1234)
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

kmeans_fit %>%
  extract_centroids()
```

`extract_cluster_assignment` *Extract cluster assignments from model*

Description

Extract cluster assignments from model

Usage

```
extract_cluster_assignment(object, ...)
```

Arguments

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

Value

A `tibble::tibble()` with 1 column `.cluster`.

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

kmeans_fit %>%
  extract_cluster_assignment()
```

`extract_fit_summary` *S3 method to get fitted model summary info depending on engine*

Description

S3 method to get fitted model summary info depending on engine

Usage

```
extract_fit_summary(object, ...)
```

Arguments

`object`	a fitted cluster_spec object
`...`	other arguments passed to methods

Details

The elements `cluster_names` and `cluster_assignments` will be factors.

Value

A list with various summary elements

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

kmeans_fit %>%
  extract_fit_summary()
```

---

finalize_model_tidyclust

Splice final parameters into objects

Description

The `finalize_*` functions take a list or tibble of tuning parameter values and update objects with those values.

Usage

```
finalize_model_tidyclust(x, parameters)

finalize_workflow_tidyclust(x, parameters)
```

Arguments

`x`	A recipe, `parsnip` model specification, or workflow.
`parameters`	A list or 1-row tibble of parameter values. Note that the column names of the tibble should be the `id` fields attached to `tune()`. For example, in the `Examples` section below, the model has `tune("K")`. In this case, the parameter tibble should be "K" and not "neighbors".

Value

An updated version of `x`.

Examples

```
kmeans_spec <- k_means(num_clusters = tune())
kmeans_spec

best_params <- data.frame(num_clusters = 5)
best_params

finalize_model_tidyclust(kmeans_spec, best_params)
```

`fit.cluster_spec` *Fit a Model Specification to a Data Set*

Description

`fit()` and `fit_xy()` take a model specification, translate_tidyclust the required code by substituting arguments, and execute the model fit routine.

Usage

```
## S3 method for class 'cluster_spec'
fit(object, formula, data, control = control_cluster(), ...)

## S3 method for class 'cluster_spec'
fit_xy(object, x, case_weights = NULL, control = control_cluster(), ...)
```

Arguments

`object`	An object of class `cluster_spec` that has a chosen engine (via `set_engine()`).
`formula`	An object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	Optional, depending on the interface (see Details below). A data frame containing all relevant variables (e.g. predictors, case weights, etc). Note: when needed, a *named argument* should be used.
`control`	A named list with elements `verbosity` and `catch`. See `control_cluster()`.
`...`	Not currently used; values passed here will be ignored. Other options required to fit the model should be passed using `set_engine()`.
`x`	A matrix, sparse matrix, or data frame of predictors. Only some models have support for sparse matrix input. See `modelenv::get_encoding()` for details. `x` should have column names.
`case_weights`	An optional classed vector of numeric case weights. This must return `TRUE` when `hardhat::is_case_weights()` is run on it. See `hardhat::frequency_weights()` and `hardhat::importance_weights()` for examples.

Details

`fit()` and `fit_xy()` substitute the current arguments in the model specification into the computational engine's code, check them for validity, then fit the model using the data and the engine-specific code. Different model functions have different interfaces (e.g. formula or `x/y`) and these functions translate_tidyclust between the interface used when `fit()` or `fit_xy()` was invoked and the one required by the underlying model.

When possible, these functions attempt to avoid making copies of the data. For example, if the underlying model uses a formula and `fit()` is invoked, the original data are references when the model is fit. However, if the underlying model uses something else, such as `x/y`, the formula is evaluated and the data are converted to the required format. In this case, any calls in the resulting model objects reference the temporary objects used to fit the model.

If the model engine has not been set, the model's default engine will be used (as discussed on each model page). If the `verbosity` option of `control_cluster()` is greater than zero, a warning will be produced.

If you would like to use an alternative method for generating contrasts when supplying a formula to `fit()`, set the global option `contrasts` to your preferred method. For example, you might set it to: `options(contrasts = c(unordered = "contr.helmert", ordered = "contr.poly"))`. See the help page for `stats::contr.treatment()` for more possible contrast types.

Value

A `cluster_fit` object that contains several elements:

- `spec`: The model specification object (`object` in the call to `fit`)
- `fit`: when the model is executed without error, this is the model object. Otherwise, it is a `try-error` object with the error message.
- `preproc`: any objects needed to convert between a formula and non-formula interface (such as the `terms` object)

The return value will also have a class related to the fitted model (e.g. `"_kmeans"`) before the base class of `"cluster_fit"`.

A fitted `cluster_fit` object.

See Also

`set_engine()`, `control_cluster()`, `cluster_spec`, `cluster_fit`

Examples

```
library(dplyr)

kmeans_mod <- k_means(num_clusters = 5)

using_formula <-
  kmeans_mod %>%
  set_engine("stats") %>%
  fit(~., data = mtcars)
```

```
using_x <-
  kmeans_mod %>%
  set_engine("stats") %>%
  fit_xy(x = mtcars)

using_formula
using_x
```

`get_centroid_dists` *Computes distance from observations to centroids*

Description

Computes distance from observations to centroids

Usage

```
get_centroid_dists(new_data, centroids, dist_fun = Rfast::dista)
```

Arguments

`new_data`	A data frame
`centroids`	A data frame where each row is a centroid.
`dist_fun`	A function for computing matrix-to-matrix distances. Defaults to `Rfast::dista()`

`glance.cluster_fit` *Construct a single row summary "glance" of a model, fit, or other object*

Description

This method glances the model in a tidyclust model object, if it exists.

Usage

```
## S3 method for class 'cluster_fit'
glance(x, ...)
```

Arguments

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

Value

a tibble

hier_clust *Hierarchical (Agglomerative) Clustering*

Description

`hier_clust()` defines a model that fits clusters based on a distance-based dendrogram

Usage

```
hier_clust(
  mode = "partition",
  engine = "stats",
  num_clusters = NULL,
  cut_height = NULL,
  linkage_method = "complete"
)
```

Arguments

`mode`	A single character string for the type of model. The only possible value for this model is "partition".
`engine`	A single character string specifying what computational engine to use for fitting. Possible engines are listed below. The default for this model is `"stats"`.
`num_clusters`	Positive integer, number of clusters in model (optional).
`cut_height`	Positive double, height at which to cut dendrogram to obtain cluster assignments (only used if `num_clusters` is `NULL`)
`linkage_method`	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"` (= UPGMA), `"mcquitty"` (= WPGMA), `"median"` (= WPGMC) or `"centroid"` (= UPGMC).

Value

A `hier_clust` cluster specification.

Examples

```
# Show all engines
modelenv::get_from_env("hier_clust")

hier_clust()
```

k_means *K-Means*

Description

`k_means()` defines a model that fits clusters based on distances to a number of centers.

Usage

```
k_means(mode = "partition", engine = "stats", num_clusters = NULL)
```

Arguments

`mode`	A single character string for the type of model. The only possible value for this model is "partition".
`engine`	A single character string specifying what computational engine to use for fitting. Possible engines are listed below. The default for this model is `"stats"`.
`num_clusters`	Positive integer, number of clusters in model.

Value

A `k_means` cluster specification.

Examples

```
# Show all engines
modelenv::get_from_env("k_means")

k_means()
```

min_grid.cluster_spec *Determine the minimum set of model fits*

Description

Determine the minimum set of model fits

Usage

```
## S3 method for class 'cluster_spec'
min_grid(x, grid, ...)
```

Arguments

`x`	A cluster 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.

`new_cluster_metric` *Construct a new clustering metric function*

Description

These functions provide convenient wrappers to create the one type of metric functions in celrry: clustering metrics. They add a metric-specific class to `fn`. These features are used by `cluster_metric_set()` and by `tune_cluster()` when tuning.

Usage

```
new_cluster_metric(fn, direction)
```

Arguments

`fn`	A function.
`direction`	A string. One of: • `"maximize"` • `"minimize"` • `"zero"`

Value

A `cluster_metric` object.

`predict.cluster_fit` *Model predictions*

Description

Apply a model to create different types of predictions. `predict()` can be used for all types of models and uses the "type" argument for more specificity.

Usage

```
## S3 method for class 'cluster_fit'
predict(object, new_data, type = NULL, opts = list(), ...)

## S3 method for class 'cluster_fit'
predict_raw(object, new_data, opts = list(), ...)
```

Arguments

`object`	An object of class `cluster_fit`
`new_data`	A rectangular data object, such as a data frame.
`type`	A single character value or `NULL`. Possible values are "cluster", or "raw". When `NULL`, `predict()` will choose an appropriate value based on the model's mode.
`opts`	A list of optional arguments to the underlying predict function that will be used when `type = "raw"`. The list should not include options for the model object or the new data being predicted.
`...`	Arguments to the underlying model's prediction function cannot be passed here (see `opts`).

Details

If "type" is not supplied to `predict()`, then a choice is made:

- `type = "cluster"` for clustering models

`predict()` is designed to provide a tidy result (see "Value" section below) in a tibble output format.

Value

With the exception of `type = "raw"`, the results of `predict.cluster_fit()` will be a tibble as many rows in the output as there are rows in `new_data` and the column names will be predictable.

For clustering results the tibble will have a `.pred_cluster` column.

Using `type = "raw"` with `predict.cluster_fit()` will return the unadulterated results of the prediction function.

When the model fit failed and the error was captured, the `predict()` function will return the same structure as above but filled with missing values. This does not currently work for multivariate models.

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

kmeans_fit %>%
  predict(new_data = mtcars)
```

prep_data_dist *Prepares data and distance matrices for metric calculation*

Description

Prepares data and distance matrices for metric calculation

Usage

```
prep_data_dist(object, new_data = NULL, dists = NULL, dist_fun = Rfast::Dist)
```

Arguments

`object`	A fitted cluster_spec object.
`new_data`	A dataset to calculate predictions on. If `NULL`, the trained cluster assignments from the fitted object are used.
`dists`	A distance matrix for the data. If `NULL`, distance is computed on `new_data` using the `stats::dist()` function.
`dist_fun`	A custom distance functions.

Value

A list

reconcile_clusterings_mapping *Relabels clusters to match another cluster assignment*

Description

When forcing one-to-one, the user needs to decide what to prioritize:

- "accuracy": optimize raw count of all observations with the same label across the two assignments
- "precision": optimize the average percent of each alt cluster that matches the corresponding primary cluster

Usage

```
reconcile_clusterings_mapping(
  primary,
  alternative,
  one_to_one = TRUE,
  optimize = "accuracy"
)
```

Arguments

`primary`	A vector containing cluster labels, to be matched
`alternative`	Another vector containing cluster labels, to be changed
`one_to_one`	Boolean; should each alt cluster match only one primary cluster?
`optimize`	One of "accuracy" or "precision"; see description.

Details

Retains the cluster labels of the primary assignment, and relabel the alternate assignment to match as closely as possible. The user must decide whether clusters are forced to be "one-to-one"; that is, are we allowed to assign multiple labels from the alternate assignment to the same primary label?

Value

A tibble with 3 columns; `primary`, `alt`, `alt_recoded`

Examples

```
factor1 <- c("Apple", "Apple", "Carrot", "Carrot", "Banana", "Banana")
factor2 <- c("Dog", "Dog", "Cat", "Dog", "Fish", "Fish")
reconcile_clusterings_mapping(factor1, factor2)

factor1 <- c("Apple", "Apple", "Carrot", "Carrot", "Banana", "Banana")
factor2 <- c("Dog", "Dog", "Cat", "Dog", "Fish", "Parrot")
reconcile_clusterings_mapping(factor1, factor2, one_to_one = FALSE)
```

`set_args.cluster_spec` *Change arguments of a cluster specification*

Description

Change arguments of a cluster specification

Usage

```
## S3 method for class 'cluster_spec'
set_args(object, ...)
```

Arguments

`object`	A model specification.
`...`	One or more named model arguments.

Value

An updated `cluster_spec` object.

set_engine.cluster_spec *Change engine of a cluster specification*

Description

Change engine of a cluster specification

Usage

```
## S3 method for class 'cluster_spec'
set_engine(object, engine, ...)
```

Arguments

object	A model specification.
engine	A character string for the software that should be used to fit the model. This is highly dependent on the type of model (e.g. linear regression, random forest, etc.).
...	Any optional arguments associated with the chosen computational engine. These are captured as quosures and can be tuned with `tune()`.

Value

An updated `cluster_spec` object.

set_mode.cluster_spec *Change mode of a cluster specification*

Description

Change mode of a cluster specification

Usage

```
## S3 method for class 'cluster_spec'
set_mode(object, mode)
```

Arguments

object	A model specification.
mode	A character string for the model type (e.g. "classification" or "regression")

Value

An updated `cluster_spec` object.

`silhouette` *Measures silhouette between clusters*

Description

Measures silhouette between clusters

Usage

```
silhouette(object, new_data = NULL, dists = NULL, dist_fun = Rfast::Dist)
```

Arguments

`object`	A fitted tidyclust model
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dists`	A distance matrix. Used if `new_data` is `NULL`.
`dist_fun`	A function for calculating distances between observations. Defaults to Euclidean distance on processed data.

Details

`silhouette_avg()` is the corresponding cluster metric function that returns the average of the values given by `silhouette()`.

Value

A tibble giving the silhouette for each observation.

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

dists <- mtcars %>%
  as.matrix() %>%
  dist()

silhouette(kmeans_fit, dists = dists)
```

silhouette_avg *Measures average silhouette across all observations*

Description

Measures average silhouette across all observations

Usage

```
silhouette_avg(object, ...)

## S3 method for class 'cluster_fit'
silhouette_avg(object, new_data = NULL, dists = NULL, dist_fun = NULL, ...)

## S3 method for class 'workflow'
silhouette_avg(object, new_data = NULL, dists = NULL, dist_fun = NULL, ...)

silhouette_avg_vec(
  object,
  new_data = NULL,
  dists = NULL,
  dist_fun = Rfast::Dist,
  ...
)
```

Arguments

`object`	A fitted kmeans tidyclust model
`...`	Other arguments passed to methods.
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dists`	A distance matrix. Used if `new_data` is `NULL`.
`dist_fun`	A function for calculating distances between observations. Defaults to Euclidean distance on processed data.

Details

Not to be confused with `silhouette()` that returns a tibble with silhouette for each observation.

Value

A double; the average silhouette.

See Also

Other cluster metric: `sse_ratio()`, `sse_total()`, `sse_within_total()`

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

dists <- mtcars %>%
  as.matrix() %>%
  dist()

silhouette_avg(kmeans_fit, dists = dists)

silhouette_avg_vec(kmeans_fit, dists = dists)
```

`sse_ratio` *Compute the ratio of the WSS to the total SSE*

Description

Compute the ratio of the WSS to the total SSE

Usage

```
sse_ratio(object, ...)

## S3 method for class 'cluster_fit'
sse_ratio(object, new_data = NULL, dist_fun = NULL, ...)

## S3 method for class 'workflow'
sse_ratio(object, new_data = NULL, dist_fun = NULL, ...)

sse_ratio_vec(object, new_data = NULL, dist_fun = Rfast::dista, ...)
```

Arguments

`object`	A fitted kmeans tidyclust model
`...`	Other arguments passed to methods.
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dist_fun`	A function for calculating distances to centroids. Defaults to Euclidean distance on processed data.

Value

A tibble with 3 columns; `.metric`, `.estimator`, and `.estimate`.

See Also

Other cluster metric: `silhouette_avg()`, `sse_total()`, `sse_within_total()`

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

sse_ratio(kmeans_fit)

sse_ratio_vec(kmeans_fit)
```

sse_total *Compute the total sum of squares*

Description

Compute the total sum of squares

Usage

```
sse_total(object, ...)

## S3 method for class 'cluster_fit'
sse_total(object, new_data = NULL, dist_fun = NULL, ...)

## S3 method for class 'workflow'
sse_total(object, new_data = NULL, dist_fun = NULL, ...)

sse_total_vec(object, new_data = NULL, dist_fun = Rfast::dista, ...)
```

Arguments

`object`	A fitted kmeans tidyclust model
`...`	Other arguments passed to methods.
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dist_fun`	A function for calculating distances to centroids. Defaults to Euclidean distance on processed data.

Value

A tibble with 3 columns; `.metric`, `.estimator`, and `.estimate`.

See Also

Other cluster metric: `silhouette_avg()`, `sse_ratio()`, `sse_within_total()`

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

sse_total(kmeans_fit)

sse_total_vec(kmeans_fit)
```

`sse_within` *Calculates Sum of Squared Error in each cluster*

Description

Calculates Sum of Squared Error in each cluster

Usage

```
sse_within(object, new_data = NULL, dist_fun = Rfast::dista)
```

Arguments

`object`	A fitted kmeans tidyclust model
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dist_fun`	A function for calculating distances to centroids. Defaults to Euclidean distance on processed data.

Details

`sse_within_total()` is the corresponding cluster metric function that returns the sum of the values given by `sse_within()`.

Value

A tibble with two columns, the cluster name and the SSE within that cluster.

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)

sse_within(kmeans_fit)
```

sse_within_total *Compute the sum of within-cluster SSE*

Description

Compute the sum of within-cluster SSE

Usage

```
sse_within_total(object, ...)

## S3 method for class 'cluster_fit'
sse_within_total(object, new_data = NULL, dist_fun = NULL, ...)

## S3 method for class 'workflow'
sse_within_total(object, new_data = NULL, dist_fun = NULL, ...)

sse_within_total_vec(object, new_data = NULL, dist_fun = Rfast::dista, ...)
```

Arguments

`object`	A fitted kmeans tidyclust model
`...`	Other arguments passed to methods.
`new_data`	A dataset to predict on. If `NULL`, uses trained clustering.
`dist_fun`	A function for calculating distances to centroids. Defaults to Euclidean distance on processed data.

Details

Not to be confused with `sse_within()` that returns a tibble with within-cluster SSE, one row for each cluster.

Value

A tibble with 3 columns; `.metric`, `.estimator`, and `.estimate`.

See Also

Other cluster metric: `silhouette_avg()`, `sse_ratio()`, `sse_total()`

Examples

```
kmeans_spec <- k_means(num_clusters = 5) %>%
  set_engine("stats")

kmeans_fit <- fit(kmeans_spec, ~., mtcars)
```

```
sse_within_total(kmeans_fit)

sse_within_total_vec(kmeans_fit)
```

`tidy.cluster_fit` *Turn a tidyclust model object into a tidy tibble*

Description

This method tidies the model in a tidyclust model object, if it exists.

Usage

```
## S3 method for class 'cluster_fit'
tidy(x, ...)
```

Arguments

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

Value

a tibble

`translate_tidyclust` *Resolve a Model Specification for a Computational Engine*

Description

`translate_tidyclust()` will translate_tidyclust a model specification into a code object that is specific to a particular engine (e.g. R package). It translate tidyclust generic parameters to their counterparts.

Usage

```
translate_tidyclust(x, ...)

## Default S3 method:
translate_tidyclust(x, engine = x$engine, ...)
```

Arguments

`x`	A model specification.
`...`	Not currently used.
`engine`	The computational engine for the model (see `?set_engine`).

Details

`translate_tidyclust()` produces a *template* call that lacks the specific argument values (such as `data`, etc). These are filled in once `fit()` is called with the specifics of the data for the model. The call may also include `tune()` arguments if these are in the specification. To handle the `tune()` arguments, you need to use the tune package. For more information see `https://www.tidymodels.org/start/tuning/`

It does contain the resolved argument names that are specific to the model fitting function/engine.

This function can be useful when you need to understand how `tidyclust` goes from a generic model specific to a model fitting function.

Note: this function is used internally and users should only use it to understand what the underlying syntax would be. It should not be used to modify the cluster specification.

Value

Prints translated code.

`tune_cluster` *Model tuning via grid search*

Description

`tune_cluster()` computes a set of performance metrics (e.g. accuracy or RMSE) for a pre-defined set of tuning parameters that correspond to a model or recipe across one or more resamples of the data.

Usage

```
tune_cluster(object, ...)

## S3 method for class 'cluster_spec'
tune_cluster(
  object,
  preprocessor,
  resamples,
  ...,
  param_info = NULL,
  grid = 10,
  metrics = NULL,
  control = tune::control_grid()
)

## S3 method for class 'workflow'
tune_cluster(
  object,
  resamples,
  ...,
  param_info = NULL,
  grid = 10,
  metrics = NULL,
  control = tune::control_grid()
)
```

Arguments

`object`	A `tidyclust` model specification or a `workflows::workflow()`.
`...`	Not currently used.
`preprocessor`	A traditional model formula or a recipe created using `recipes::recipe()`.
`resamples`	An `rset()` object.
`param_info`	A `dials::parameters()` object or `NULL`. If none is given, a parameters set is derived from other arguments. Passing this argument can be useful when parameter ranges need to be customized.
`grid`	A data frame of tuning combinations or a positive integer. The data frame should have columns for each parameter being tuned and rows for tuning parameter candidates. An integer denotes the number of candidate parameter sets to be created automatically.
`metrics`	A `cluster_metric_set()` or `NULL`.
`control`	An object used to modify the tuning process. Defaults to `tune::control_grid()`.

Value

An updated version of `resamples` with extra list columns for `.metrics` and `.notes` (optional columns are `.predictions` and `.extracts`). `.notes` contains warnings and errors that occur during execution.

Examples

```
library(recipes)
library(rsample)
library(workflows)
library(tune)

rec_spec <- recipe(~., data = mtcars) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pca(all_numeric_predictors())

kmeans_spec <- k_means(num_clusters = tune())

wflow <- workflow() %>%
  add_recipe(rec_spec) %>%
  add_model(kmeans_spec)

grid <- tibble(num_clusters = 1:3)

set.seed(4400)
folds <- vfold_cv(mtcars, v = 2)
```

```
res <- tune_cluster(
  wflow,
  resamples = folds,
  grid = grid
)
res

collect_metrics(res)
```

update.hier_clust *Update a cluster specification*

Description

If parameters of a cluster specification need to be modified, `update()` can be used in lieu of recreating the object from scratch.

Usage

```
## S3 method for class 'hier_clust'
update(
  object,
  parameters = NULL,
  num_clusters = NULL,
  cut_height = NULL,
  linkage_method = NULL,
  fresh = FALSE,
  ...
)

## S3 method for class 'k_means'
update(object, parameters = NULL, num_clusters = NULL, fresh = FALSE, ...)
```

Arguments

`object`	A cluster specification.
`parameters`	A 1-row tibble or named list with *main* parameters to update. Use **either** `parameters` **or** the main arguments directly when updating. If the main arguments are used, these will supersede the values in `parameters`. Also, using engine arguments in this object will result in an error.
`num_clusters`	Positive integer, number of clusters in model.
`cut_height`	Positive double, height at which to cut dendrogram to obtain cluster assignments (only used if `num_clusters` is `NULL`)
`linkage_method`	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"` (= UPGMA), `"mcquitty"` (= WPGMA), `"median"` (= WPGMC) or `"centroid"` (= UPGMC).

`fresh`	A logical for whether the arguments should be modified in-place or replaced wholesale.
`...`	Not used for `update()`.

Value

An updated cluster specification.

Examples

```
kmeans_spec <- k_means(num_clusters = 5)
kmeans_spec
update(kmeans_spec, num_clusters = 1)
update(kmeans_spec, num_clusters = 1, fresh = TRUE)

param_values <- tibble::tibble(num_clusters = 10)

kmeans_spec %>% update(param_values)
```

Index