Package 'mlr3cluster'

October 17, 2026

Title Cluster Extension for 'mlr3'

Version 0.7.0

Description Extends the 'mlr3' package with cluster analysis.

License LGPL-3

URL https://mlr3cluster.mlr-org.com, https://github.com/mlr-org/mlr3cluster

BugReports https://github.com/mlr-org/mlr3cluster/issues

Depends mlr3 (>= 1.6.0), R (>= 3.5.0)

Imports backports (>= 1.5.0), checkmate (>= 2.0.0), cluster, data.table (>= 1.15.0), graphics, methods, mlr3misc (>= 0.21.0), paradox (>= 1.0.1), R6 (>= 2.4.1), stats, utils

Suggests apcluster, clue (>= 0.3-55), clusterCrit, ClusterR (>= 1.3.3), clustMixType (>= 0.4-2), dbscan (>= 1.2.1), e1071, flexclust, flexmix, fpc, future, genieclust, gmeans, kernlab, klaR (>= 0.6-13), kohonen (>= 3.0.10), lgr, LPCM, mclust (>= 5.3), mirai (>= 2.4.1), movMF (>= 0.1-1), mvtnorm, protoclust, rJava, RWeka, skmeans (>= 0.2-0), stdbscan (>= 0.2.0), stream (>= 2.0.0), tclust (>= 2.2-3), testthat (>= 3.3.0), withr

Config/roxygen2/markdown TRUE

Config/roxygen2/r6 TRUE

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Encoding UTF-8

Collate 'LearnerClust.R' 'zzz.R' 'LearnerClustAffinityPropagation.R' 'LearnerClustAgnes.R' 'LearnerClustBICO.R' 'LearnerClustBIRCH.R' 'LearnerClustCLARA.R' 'LearnerClustCMeans.R' 'LearnerClustCobweb.R' 'LearnerClustDBSCAN.R' 'LearnerClustDBSCANfpc.R' 'LearnerClustDiana.R' 'LearnerClustEM.R' 'LearnerClustFanny.R' 'LearnerClustFarthestFirst.R' 'LearnerClustFeatureless.R' 'LearnerClustFlexmix.R' 'LearnerClustGMeans.R'

'LearnerClustGenie.R' 'LearnerClustHDBSCAN.R'
'LearnerClustHclust.R' 'LearnerClustKCCA.R'
'LearnerClustKKMeans.R' 'LearnerClustKMeans.R'
'LearnerClustKMeansRcpp.R' 'LearnerClustKModes.R'
'LearnerClustKProto.R' 'LearnerClustMclust.R'
'LearnerClustMeanShift.R' 'LearnerClustMiniBatchKMeans.R'
'LearnerClustMovMF.R' 'LearnerClustOPTICS.R'
'LearnerClustPAM.R' 'LearnerClustProtoclust.R'
'LearnerClustSKMeans.R' 'LearnerClustSOM.R'
'LearnerClustSTDBSCAN.R' 'LearnerClustSimpleKMeans.R'
'LearnerClustSpectral.R' 'LearnerClustTclust.R'
'LearnerClustXMeans.R' 'MeasureClust.R' 'measures.R'
'cluster_stats.R' 'MeasureClustSimple.R' 'PredictionClust.R'
'PredictionDataClust.R' 'TaskClust.R' 'TaskClust_ruspini.R'
'TaskClust_usarrest.R' 'TaskGeneratorBlobs.R'
'as_prediction_clust.R' 'as_task_clust.R' 'bibentries.R'
'default_fallback.R' 'helper.R' 'helper_weka.R'

NeedsCompilation no

Author Maximilian Mücke [aut, cre] (ORCID:
<https://orcid.org/0009-0000-9432-9795>),
Damir Pulatov [aut],
Michel Lang [aut] (ORCID: <https://orcid.org/0000-0001-9754-0393>),
Marc Becker [ctb] (ORCID: <https://orcid.org/0000-0002-8115-0400>)

Maintainer Maximilian Mücke <muecke.maximilian@gmail.com>

Repository CRAN

Date/Publication 2026-10-17 19:50:02 UTC

Contents

mlr3cluster-package 4
as_prediction_clust 4
as_task_clust 5
LearnerClust 7
MeasureClust 10
mlr_learners_clust.agnes 12
mlr_learners_clust.ap 15
mlr_learners_clust.bico 18
mlr_learners_clust.birch 20
mlr_learners_clust.clara 22
mlr_learners_clust.cmeans 25
mlr_learners_clust.cobweb 28
mlr_learners_clust.dbscan 31
mlr_learners_clust.dbscan_fpc 33
mlr_learners_clust.diana 35
mlr_learners_clust.em 38
mlr_learners_clust.fanny 41

mlr_learners_clust.featureless . . . 44
mlr_learners_clust.ff . . . 46
mlr_learners_clust.flexmix . . . 49
mlr_learners_clust.genie . . . 52
mlr_learners_clust.gmeans . . . 54
mlr_learners_clust.hclust . . . 57
mlr_learners_clust.hdbscan . . . 60
mlr_learners_clust.kcca . . . 62
mlr_learners_clust.kkmeans . . . 65
mlr_learners_clust.kmeans . . . 68
mlr_learners_clust.kmeans_rcpp . . . 70
mlr_learners_clust.kmodes . . . 73
mlr_learners_clust.kproto . . . 76
mlr_learners_clust.MBatchKMeans . . . 79
mlr_learners_clust.mclust . . . 81
mlr_learners_clust.meanshift . . . 84
mlr_learners_clust.movMF . . . 87
mlr_learners_clust.optics . . . 90
mlr_learners_clust.pam . . . 92
mlr_learners_clust.protoclust . . . 95
mlr_learners_clust.SimpleKMeans . . . 98
mlr_learners_clust.skmeans . . . 101
mlr_learners_clust.som . . . 104
mlr_learners_clust.specc . . . 107
mlr_learners_clust.stdbscan . . . 109
mlr_learners_clust.tclust . . . 112
mlr_learners_clust.xmeans . . . 115
mlr_measures_clust.avg_between . . . 118
mlr_measures_clust.avg_within . . . 119
mlr_measures_clust.ch . . . 120
mlr_measures_clust.davies_bouldin . . . 121
mlr_measures_clust.dunn . . . 122
mlr_measures_clust.dunn2 . . . 123
mlr_measures_clust.entropy . . . 124
mlr_measures_clust.pearsongamma . . . 125
mlr_measures_clust.silhouette . . . 126
mlr_measures_clust.sse_ratio . . . 127
mlr_measures_clust.wb_ratio . . . 128
mlr_measures_clust.wss . . . 129
mlr_tasks_ruspini . . . 130
mlr_tasks_usarrests . . . 131
mlr_task_generators_blobs . . . 132
PredictionClust . . . 134
TaskClust . . . 136

Index **138**

mlr3cluster-package *mlr3cluster: Cluster Extension for 'mlr3'*

Description

Extends the 'mlr3' package with cluster analysis.

Author(s)

Maintainer: Maximilian Mücke <muecke.maximilian@gmail.com> (ORCID)

Authors:

- Maximilian Mücke <muecke.maximilian@gmail.com> (ORCID)
- Damir Pulatov <damirpolat@protonmail.com>
- Michel Lang <michellang@gmail.com> (ORCID)

Other contributors:

- Marc Becker <marcbecker@posteo.de> (ORCID) [contributor]

See Also

Useful links:

- https://mlr3cluster.mlr-org.com
- https://github.com/mlr-org/mlr3cluster
- Report bugs at https://github.com/mlr-org/mlr3cluster/issues

as_prediction_clust *Convert to a Cluster Prediction*

Description

Convert object to a PredictionClust. For a `data.frame`, the columns `row_ids` and `partition` are required. Columns prefixed with `prob.` are collected into the probability matrix, `weights` are stored as measure weights, and all remaining columns are stored as extra data in the `extra` field of the prediction.

Usage

```
as_prediction_clust(x, ...)

## S3 method for class 'PredictionClust'
as_prediction_clust(x, ...)

## S3 method for class 'data.frame'
as_prediction_clust(x, ...)
```

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.

Value

PredictionClust.

Examples

```
# create a prediction object
task = tsk("usarrests")
learner = lrn("clust.cmeans", predict_type = "prob")
learner$train(task)
p = learner$predict(task)

# convert to a data.table
tab = as.data.table(p)

# convert back to a Prediction
as_prediction_clust(tab)

# split data.table into a 3 data.tables based on UrbanPop
f = cut(task$data(rows = tab$row_ids)$UrbanPop, 3)
tabs = split(tab, f)

# convert back to list of predictions
preds = lapply(tabs, as_prediction_clust)

# calculate performance in each group
sapply(preds, function(p) p$score(task = task))
```

---

as_task_clust *Convert to a Cluster Task*

Description

Convert object to a TaskClust or a list of TaskClust. This is a S3 generic, specialized for at least the following objects:

1. TaskClust: returns the object as-is, possibly cloned.
2. `formula`, `data.frame()`, `matrix()`, and mlr3::DataBackend: provides an alternative to the constructor of TaskClust.
3. mlr3::TaskSupervised (e.g., mlr3::TaskClassif or mlr3::TaskRegr): converts the task to a TaskClust by dropping the target, analogous to `mlr3::convert_task()`.

Usage

```
as_task_clust(x, ...)

## S3 method for class 'TaskClust'
as_task_clust(x, clone = FALSE, ...)

## S3 method for class 'TaskSupervised'
as_task_clust(x, drop_levels = TRUE, ...)

## S3 method for class 'data.frame'
as_task_clust(x, id = deparse1(substitute(x)), label = NA_character_, ...)

## S3 method for class 'matrix'
as_task_clust(x, id = deparse1(substitute(x)), label = NA_character_, ...)

## S3 method for class 'DataBackend'
as_task_clust(x, id = deparse1(substitute(x)), label = NA_character_, ...)

## S3 method for class 'formula'
as_task_clust(
  x,
  data,
  id = deparse1(substitute(data)),
  label = NA_character_,
  ...
)

as_tasks_clust(x, ...)

## S3 method for class 'list'
as_tasks_clust(x, clone = FALSE, ...)

## S3 method for class 'TaskClust'
as_tasks_clust(x, clone = FALSE, ...)
```

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.
clone	(logical(1)) If TRUE, ensures that the returned object is not the same as the input x.
drop_levels	(logical(1)) If TRUE, drops unused levels of factor columns, as in mlr3::convert_task().
id	(character(1)) Id for the new task. Defaults to the (deparsed and substituted) name of the data

argument.

`label` (`character(1)`)
Label for the new instance.

`data` (`data.frame()`)
Data frame containing all columns specified in formula `x`.

Value

TaskClust.

Examples

```
as_task_clust(datasets::USArrests)

# convert a classification task to a cluster task
as_task_clust(tsk("iris"))

# convert a generated classification task to a cluster task
as_task_clust(tgen("moons")$generate(100))
```

LearnerClust *Cluster Learner*

Description

This Learner specializes mlr3::Learner for cluster problems:

- `task_type` is set to `"clust"`.
- Creates mlr3::Predictions of class PredictionClust.
- Possible values for `predict_types` are:
  - `"partition"`: Integer indicating the cluster membership.
  - `"prob"`: Probability for belonging to each cluster.
- Additional learner properties include:
  - `"exclusive"`: The method natively assigns each observation to exactly one cluster.
  - `"overlapping"`: The method natively assigns observations to multiple clusters.
  - `"fuzzy"`: The method natively produces soft cluster memberships, e.g. fuzzy or probabilistic model-based methods. The hard partition is derived from the memberships.
  - `"complete"`: Every observation is assigned to a cluster.
  - `"partial"`: Observations may be left unassigned, e.g. as noise points.
  - `"partitional"`: The method divides the data into non-nested clusters.
  - `"hierarchical"`: The method produces a nested hierarchy of clusters.
  - `"density"`: The method finds clusters as dense regions in the feature space.

  These properties describe the nature of the underlying method, not its interface capabilities: whether a learner can return soft memberships is encoded by the `"prob"` predict type, which `"exclusive"` learners may also support via derived scores.

Predefined learners can be found in the mlr3misc::Dictionary mlr3::mlr_learners.

Super class

`mlr3::Learner` -> `LearnerClust`

Public fields

`assignments` (`NULL` | `vector()`)
Cluster assignments from learned model.

Active bindings

`save_assignments` (`logical(1)`)
Should assignments for 'train' data be saved in the learner? Default is `TRUE`.

Methods

Public methods:

- `LearnerClust$new()`
- `LearnerClust$reset()`
- `LearnerClust$clone()`

`LearnerClust$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClust$new(
  id,
  param_set = ps(),
  predict_types = "partition",
  feature_types = character(),
  properties = character(),
  packages = character(),
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id` (`character(1)`)
Identifier for the new instance.

`param_set` (paradox::ParamSet)
Set of hyperparameters.

`predict_types` (`character()`)
Supported predict types. Must be a subset of `mlr_reflections$learner_predict_types`.

`feature_types` (`character()`)
Feature types the learner operates on. Must be a subset of `mlr_reflections$task_feature_types`.

`properties` (`character()`)
Set of properties of the mlr3::Learner. Must be a subset of `mlr_reflections$learner_properties`.
The following properties are currently standardized and understood by learners in **mlr3**:

- `"missings"`: The learner can handle missing values in the data.
- `"weights"`: The learner supports observation weights.

- `"offset"`: The learner can incorporate offset values to adjust predictions.
- `"importance"`: The learner supports extraction of importance scores, i.e. comes with an `$importance()` extractor function (see section on optional extractors in mlr3::Learner).
- `"selected_features"`: The learner supports extraction of the set of selected features, i.e. comes with a `$selected_features()` extractor function (see section on optional extractors in mlr3::Learner).
- `"oob_error"`: The learner supports extraction of estimated out of bag error, i.e. comes with a `oob_error()` extractor function (see section on optional extractors in mlr3::Learner).
- `"validation"`: The learner can use a validation task during training.
- `"internal_tuning"`: The learner is able to internally optimize hyperparameters (those are also tagged with `"internal_tuning"`).
- `"marshal"`: To save learners with this property, you need to call `$marshal()` first. If a learner is in a marshaled state, you call first need to call `$unmarshal()` to use its model, e.g. for prediction.
- `"hotstart_forward"`: The learner supports to hotstart a model forward.
- `"hotstart_backward"`: The learner supports hotstarting a model backward.
- `"featureless"`: The learner does not use features.

`packages (character())`
: Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via `requireNamespace()`.

`label (character(1))`
: Label for the new instance.

`man (character(1))`
: String in the format `[pkg]::[topic]` pointing to a manual page for this object. The referenced help package can be opened via method `$help()`.

`LearnerClust$reset()`: Reset `assignments` field before calling parent's `reset()`.

Usage:

```
LearnerClust$reset()
```

`LearnerClust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(mlr3)
library(mlr3cluster)
ids = mlr_learners$keys("^clust")
ids

# get a specific learner from mlr_learners:
learner = lrn("clust.featureless")
print(learner)
```

MeasureClust *Cluster Measure*

Description

This measure specializes mlr3::Measure for cluster analysis:

- `task_type` is set to `"clust"`.
- Possible values for `predict_type` are `"partition"` and `"prob"`.

Predefined measures can be found in the mlr3misc::Dictionary mlr3::mlr_measures.

Super class

`mlr3::Measure` -> `MeasureClust`

Methods

Public methods:

- `MeasureClust$new()`
- `MeasureClust$clone()`

`MeasureClust$new()`: Creates a new instance of this R6 class.

Usage:

```
MeasureClust$new(
  id,
  param_set = ps(),
  range,
  minimize = NA,
  average = "macro",
  aggregator = NULL,
  properties = character(),
  predict_type = "partition",
  predict_sets = "test",
  task_properties = character(),
  packages = character(),
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id` (`character(1)`)
Identifier for the new instance.

`param_set` (paradox::ParamSet)
Set of hyperparameters.

`range` (`numeric(2)`)
: Feasible range for this measure as `c(lower_bound, upper_bound)`. Both bounds may be infinite.

`minimize` (`logical(1)`)
: Set to `TRUE` if good predictions correspond to small values, and to `FALSE` if good predictions correspond to large values. If set to `NA` (default), tuning this measure is not possible.

`average` (`character(1)`)
: How to average multiple mlr3::Predictions from a ResampleResult.
The default, `"macro"`, calculates the individual performances scores for each mlr3::Prediction and then uses the function defined in `$aggregator` to average them to a single number.
`"macro_weighted"` is similar to `"macro"`, but uses weighted averages. Weights are taken from the `weights_measure` column of the resampled mlr3::Task if present. Note that `"macro_weighted"` can differ from `"macro"` even if no weights are present or if `$use_weights` is set to `"ignore"`, since then aggregation is done using *uniform sample weights*, which result in non-uniform weights for mlr3::Predictions if they contain different numbers of samples.
If set to `"micro"`, the individual mlr3::Prediction objects are first combined into a single new mlr3::Prediction object which is then used to assess the performance. The function in `$aggregator` is not used in this case.

`aggregator` (`function()` | `NULL`)
: Function to aggregate over multiple iterations. The role of this function depends on the value of field `"average"`:
  - `"macro"`: A numeric vector of scores (one per iteration) is passed. The aggregate function defaults to `mean()` in this case.
  - `"micro"`: The `aggregator` function is not used. Instead, predictions from multiple iterations are first combined and then scored in one go.
  - `"custom"`: A ResampleResult is passed to the aggregate function.

`properties` (`character()`)
: Properties of the measure. Must be a subset of mlr_reflections$measure_properties. Supported by `mlr3`:
  - `"requires_task"` (requires the complete mlr3::Task),
  - `"requires_learner"` (requires the trained mlr3::Learner),
  - `"requires_model"` (requires the trained mlr3::Learner, including the fitted model),
  - `"requires_train_set"` (requires the training indices from the mlr3::Resampling),
  - `"na_score"` (the measure is expected to occasionally return `NA` or `NaN`),
  - `"weights"` (support weighted scoring using sample weights from task, column role `weights_measure`),
  - `"primary_iters"` (the measure explicitly handles resamplings that only use a subset of their iterations for the point estimate), and
  - `"requires_no_prediction"` (No prediction is required; This usually means that the measure extracts some information from the learner state.).

`predict_type` (`character(1)`)
: Required predict type of the mlr3::Learner. Possible values are stored in mlr_reflections$learner_predict_types.

`predict_sets` (`character()`)
: Prediction sets to operate on, used in `aggregate()` to extract the matching `predict_sets` from the ResampleResult. Multiple predict sets are calculated by the respective mlr3::Learner during `resample()`/`benchmark()`. Must be a non-empty subset of `{"train", "test", "internal_valid"}`.

If multiple sets are provided, these are first combined to a single prediction object. Default is "test".

task_properties (character())
: Required task properties, see mlr3::Task.

packages (character())
: Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().

label (character(1))
: Label for the new instance.

man (character(1))
: String in the format [pkg]::[topic] pointing to a manual page for this object. The referenced help package can be opened via method $help().

MeasureClust$clone(): The objects of this class are cloneable with this method.

Usage:

```
MeasureClust$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Example cluster measures: clust.dunn

---

mlr_learners_clust.agnes

Agglomerative Nesting Clustering Learner

Description

Agglomerative hierarchical clustering. Calls cluster::agnes() from package **cluster**.

The predict method uses stats::cutree() which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter k).

Custom mlr3 parameters

- k:
  - Not an argument of cluster::agnes(). The number of clusters to cut the tree into, passed to stats::cutree(). Initialized to 2.

Initial parameter values

- `keep.diss`:
  - Actual default: `n < 100`, where `n` is the number of observations.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the dissimilarity matrix in the model to save memory.
- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.agnes")
lrn("clust.agnes")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

Parameters

Id	Type	Default	Levels	Range
metric	character	euclidean	euclidean, manhattan	-
stand	logical	FALSE	TRUE, FALSE	-
method	character	average	average, single, complete, ward, weighted, flexible, gaverage	-
keep.diss	logical	-	TRUE, FALSE	-
keep.data	logical	TRUE	TRUE, FALSE	-
trace.lev	integer	0		$[0, \infty)$
k	integer	-		$[1, \infty)$
par.method	untyped	-		-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustAgnes`

Methods

Public methods:

- `LearnerClustAgnes$new()`
- `LearnerClustAgnes$clone()`

`LearnerClustAgnes$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustAgnes$new()
```

`LearnerClustAgnes$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustAgnes$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Kaufman, Leonard, Rousseeuw, J P (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`, `mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbscan_fpc`, `mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele` `mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kcca`, `mlr_learners_clust.kkme` `mlr_learners_clust.kmeans`, `mlr_learners_clust.kmeans_rcpp`, `mlr_learners_clust.kmodes`, `mlr_learners_clust.kproto`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.movMF`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.protocl` `mlr_learners_clust.skmeans`, `mlr_learners_clust.som`, `mlr_learners_clust.specc`, `mlr_learners_clust.stdbsc` `mlr_learners_clust.tclust`, `mlr_learners_clust.xmeans`

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.agnes")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.ap *Affinity Propagation Clustering Learner*

Description

Affinity Propagation clustering. Calls `apcluster::apcluster()` from package **apcluster**.

Note that `apcluster::apcluster()` doesn't have a default for the similarity function. The predict method computes the closest cluster exemplar to find the cluster memberships for new data. The code is taken from StackOverflow answer by the `apcluster` package maintainer.

The similarity `s` can be a function, e.g. `apcluster::negDistMat(r = 2)`, or the name of a function such as `"negDistMat"`. For a function without a `sel` argument, predicting computes the full similarity matrix of the exemplars and the new data.

Initial parameter values

- `includeSim`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the n x n similarity matrix in the model.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.ap")
lrn("clust.ap")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **apcluster**

Parameters

Id	Type	Default	Levels	Range
s	untyped	-		-
p	untyped	NA_real_		-
q	numeric	NA		$[0, 1]$
maxits	integer	1000		$[1, \infty)$
convits	integer	100		$[1, \infty)$
lam	numeric	0.9		$[0.5, 1]$
includeSim	logical	TRUE	TRUE, FALSE	-
details	logical	FALSE	TRUE, FALSE	-
nonoise	logical	FALSE	TRUE, FALSE	-
seed	integer	NA		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustAP`

Active bindings

`native_model (APResult)`
The fitted model returned by `apcluster::apcluster()`.

Methods

Public methods:

- `LearnerClustAP$new()`
- `LearnerClustAP$clone()`

`LearnerClustAP$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustAP$new()
```

`LearnerClustAP$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustAP$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Bodenhofer, Ulrich, Kothmeier, Andreas, Hochreiter, Sepp (2011). "APCluster: an R package for affinity propagation clustering." *Bioinformatics*, **27**(17), 2463–2464.

Frey, J B, Dueck, Delbert (2007). "Clustering by passing messages between data points." *science*, **315**(5814), 972–976.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara, mlr_learners_clust.cmeans,
mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc,
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.ap")
print(learner)
```

`mlr_learners_clust.bico`
BICO Clustering Learner

Description

BICO (fast computation of k-means coresets in a data stream) clustering. Calls `stream::DSC_BICO()` from package **stream**.

`stream::DSC_BICO()` only computes the coreset (micro-clusters), so the coreset is reclustered with k-means via `stream::DSC_TwoStage()` and `stream::DSC_Kmeans()` to obtain the final partition with k clusters.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.bico")
lrn("clust.bico")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **stream**

Parameters

Id	Type	Default	Range
k	integer	5	$[2, \infty)$
space	integer	10	$[1, \infty)$
p	integer	10	$[1, \infty)$
iterations	integer	10	$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustBICO`

Methods

Public methods:

- `LearnerClustBICO$new()`
- `LearnerClustBICO$clone()`

`LearnerClustBICO$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustBICO$new()
```

`LearnerClustBICO$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustBICO$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Fichtenberger, Hendrik, Gille, Marc, Schmidt, Melanie, Schwiegelshohn, Chris, Sohler, Christian (2013). "BICO: BIRCH Meets Coresets for k-Means Clustering." In *Algorithms–ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings 21*, 481–492. Springer.

Hahsler M, Bolaños M, Forrest J (2017). "Introduction to stream: An Extensible Framework for Data Stream Clustering Research with R." *Journal of Statistical Software*, **76**(14), 1–50. doi:10.18637/jss.v076.i14.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners`
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`, `mlr_learners_clust.cmeans`,
`mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbscan_fpc`,

mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.bico")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.birch

BIRCH Clustering Learner

Description

BIRCH (balanced iterative reducing clustering using hierarchies) clustering. Calls `stream::DSC_BIRCH()` from package **stream**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.birch")
lrn("clust.birch")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **stream**

Parameters

Id	Type	Default	Range
threshold	numeric	-	$[0, \infty)$
branching	integer	-	$[1, \infty)$
maxLeaf	integer	-	$[1, \infty)$
maxMem	integer	0	$[0, \infty)$
outlierThreshold	numeric	0.25	$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustBIRCH`

Methods

Public methods:

- `LearnerClustBIRCH$new()`
- `LearnerClustBIRCH$clone()`

`LearnerClustBIRCH$new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClustBIRCH$new()
```

`LearnerClustBIRCH$clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LearnerClustBIRCH$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Zhang, Tian, Ramakrishnan, Raghu, Livny, Miron (1996). "BIRCH: An Efficient Data Clustering Method for Very Large Databases." *ACM sigmod record*, **25**(2), 103–114.

Zhang, Tian, Ramakrishnan, Raghu, Livny, Miron (1997). "BIRCH: A new data clustering algorithm and its applications." *Data Mining and Knowledge Discovery*, **1**, 141–182.

Hahsler M, Bolaños M, Forrest J (2017). "Introduction to stream: An Extensible Framework for Data Stream Clustering Research with R." *Journal of Statistical Software*, **76**(14), 1–50. doi:10.18637/jss.v076.i14.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.clara, mlr_learners_clust.cmeans,
mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc,
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.birch")
print(learner)
```

`mlr_learners_clust.clara`

CLARA Clustering Learner

Description

Clustering Large Applications (CLARA) clustering. Calls `cluster::clara()` from package **cluster**.

CLARA extends the PAM algorithm to handle larger datasets by working on sub-datasets of fixed size. The `k` parameter is set to 2 by default since `cluster::clara()` doesn't have a default value for the number of clusters. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Initial parameter values

- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.clara")
lrn("clust.clara")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**, **clue**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
metric	character	euclidean	euclidean, manhattan, jaccard	-
stand	logical	FALSE	TRUE, FALSE	-
samples	integer	5		$[1, \infty)$
sampsize	integer	-		$[1, \infty)$
trace	integer	0		$[0, \infty)$
medoids.x	logical	TRUE	TRUE, FALSE	-
keep.data	logical	TRUE	TRUE, FALSE	-
rngR	logical	FALSE	TRUE, FALSE	-
pamLike	logical	FALSE	TRUE, FALSE	-
correct.d	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustCLARA`

Methods

Public methods:

- `LearnerClustCLARA$new()`
- `LearnerClustCLARA$clone()`

`LearnerClustCLARA$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustCLARA$new()
```

`LearnerClustCLARA$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustCLARA$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Kaufman, Leonard, Rousseeuw, J P (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

Schubert, Erich, Rousseeuw, J P (2019). "Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms." In *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings 12*, 171–187. Springer.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.cmeans`,
`mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbscan_fpc`,
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,

mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbs
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.clara")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.cmeans

Fuzzy C-Means Clustering Learner

Description

Fuzzy c-means clustering. Calls `e1071::cmeans()` from package **e1071**.

The `centers` parameter is set to 2 by default since `e1071::cmeans()` doesn't have a default value for the number of clusters. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.cmeans")
lrn("clust.cmeans")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3cluster**, **e1071**, **clue**

Parameters

Id	Type	Default	Levels	Range
centers	untyped	-		-
iter.max	integer	100		$[1, \infty)$
verbose	logical	FALSE	TRUE, FALSE	-
dist	character	euclidean	euclidean, manhattan	-
method	character	cmeans	cmeans, ufcl	-
m	numeric	2		$[1, \infty)$
rate.par	numeric	-		$[0, 1]$
weights	untyped	1L		-
control	untyped	-		-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustCMeans`

Methods

Public methods:

- `LearnerClustCMeans$new()`
- `LearnerClustCMeans$clone()`

`LearnerClustCMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustCMeans$new()
```

`LearnerClustCMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustCMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Dimitriadou, Evgenia, Hornik, Kurt, Leisch, Friedrich, Meyer, David, Weingessel, Andreas (2008). "Misc functions of the Department of Statistics (e1071), TU Wien." *R package*, **1**, 5–24.

Bezdek, C J (2013). *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc,
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.cmeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
```

```
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.cobweb`

Cobweb Clustering Learner

Description

Cobweb clustering. Calls `RWeka::Cobweb()` from package **RWeka**.

The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.cobweb")
lrn("clust.cobweb")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3cluster**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
A	numeric	1		$[0, \infty)$
C	numeric	0.002820948		$[0, \infty)$
save_data	logical	FALSE	TRUE, FALSE	-
S	integer	42		$[0, \infty)$
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustCobweb`

Active bindings

`marshaled (logical(1))`
Whether the learner's model is marshaled.

Methods

Public methods:

- `LearnerClustCobweb$new()`
- `LearnerClustCobweb$marshal()`
- `LearnerClustCobweb$unmarshal()`
- `LearnerClustCobweb$clone()`

`LearnerClustCobweb$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustCobweb$new()
```

`LearnerClustCobweb$marshal()`: Marshal the learner's model.

Usage:

```
LearnerClustCobweb$marshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::marshal_model()`.

`LearnerClustCobweb$unmarshal()`: Unmarshal the learner's model.

Usage:

```
LearnerClustCobweb$unmarshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::unmarshal_model()`.

`LearnerClustCobweb$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustCobweb$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Witten, H I, Frank, Eibe (2002). "Data mining: practical machine learning tools and techniques with Java implementations." *Acm Sigmod Record*, **31**(1), 76–77.

Fisher, H D (1987). "Knowledge acquisition via incremental conceptual clustering." *Machine learning*, **2**, 139–172.

Gennari, H J, Langley, Pat, Fisher, Doug (1989). "Models of incremental concept formation." *Artificial intelligence*, **40**(1-3), 11–61.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc,
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.cobweb")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.dbscan`
DBSCAN Clustering Learner

Description

DBSCAN (density-based spatial clustering of applications with noise) clustering. Calls `dbscan::dbscan()` from package **dbscan**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.dbscan")
lrn("clust.dbscan")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **dbscan**

Parameters

Id	Type	Default	Levels	Range
eps	numeric	-		$[0, \infty)$
minPts	integer	5		$[0, \infty)$
weights	untyped	NULL		-
borderPoints	logical	TRUE	TRUE, FALSE	-
search	character	kdtree	kdtree, linear, dist	-
bucketSize	integer	10		$[1, \infty)$
splitRule	character	SUGGEST	STD, MIDPT, FAIR, SL_MIDPT, SL_FAIR, SUGGEST	-
approx	numeric	0		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustDBSCAN`

Active bindings

`native_model` (`dbscan_fast`)
The fitted model returned by `dbscan::dbscan()`.

Methods

Public methods:

- `LearnerClustDBSCAN$new()`
- `LearnerClustDBSCAN$clone()`

`LearnerClustDBSCAN$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustDBSCAN$new()
```

`LearnerClustDBSCAN$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustDBSCAN$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Hahsler M, Piekenbrock M, Doran D (2019). "dbscan: Fast Density-Based Clustering with R." *Journal of Statistical Software*, **91**(1), 1–30. doi:10.18637/jss.v091.i01.

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei, others (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In *kdd*, volume 96 number 34, 226–231.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan_fpc`,
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,

mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.dbscan")
print(learner)
```

`mlr_learners_clust.dbscan_fpc`

DBSCAN Clustering Learner (fpc)

Description

DBSCAN (density-based spatial clustering of applications with noise) clustering. Calls `fpc::dbscan()` from package **fpc**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.dbscan_fpc")
lrn("clust.dbscan_fpc")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **fpc**

Parameters

Id	Type	Default	Levels	Range
eps	numeric	-		$[0, \infty)$
MinPts	integer	5		$[0, \infty)$
scale	logical	FALSE	TRUE, FALSE	-
method	character	hybrid	hybrid, raw	-
seeds	logical	TRUE	TRUE, FALSE	-

showplot	untyped	FALSE	-
countmode	untyped	NULL	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustDBSCANfpc`

Active bindings

`native_model` (dbscan)
The fitted model returned by `fpc::dbscan()`.

Methods

Public methods:

- `LearnerClustDBSCANfpc$new()`
- `LearnerClustDBSCANfpc$clone()`

`LearnerClustDBSCANfpc$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustDBSCANfpc$new()
```

`LearnerClustDBSCANfpc$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustDBSCANfpc$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei, others (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In *kdd*, volume 96 number 34, 226–231.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.

- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dia
mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featureless, mlr_learners_clust.ff,
mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hcl
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.dbscan_fpc")
print(learner)
```

`mlr_learners_clust.diana`

Divisive Analysis Clustering Learner

Description

Divisive hierarchical clustering. Calls `cluster::diana()` from package **cluster**.

The predict method uses `stats::cutree()` which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter k).

Custom mlr3 parameters

- k:
  - Not an argument of `cluster::diana()`. The number of clusters to cut the tree into, passed to `stats::cutree()`. Initialized to 2.

Initial parameter values

- `keep.diss`:
  - Actual default: `n < 100`, where `n` is the number of observations.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the dissimilarity matrix in the model to save memory.
- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.diana")
lrn("clust.diana")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

Parameters

Id	Type	Default	Levels	Range
metric	character	euclidean	euclidean, manhattan	-
stand	logical	FALSE	TRUE, FALSE	-
keep.diss	logical	-	TRUE, FALSE	-
keep.data	logical	TRUE	TRUE, FALSE	-
trace.lev	integer	0		$[0, \infty)$
k	integer	-		$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustDiana`

Methods

Public methods:

- `LearnerClustDiana$new()`
- `LearnerClustDiana$clone()`

`LearnerClustDiana$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustDiana$new()
```

`LearnerClustDiana$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustDiana$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Kaufman, Leonard, Rousseeuw, J P (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featureless`, `mlr_learners_clust.ff`,
`mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`, `mlr_learners_clust.hc`
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kcca`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.kme`
`mlr_learners_clust.kmeans_rcpp`, `mlr_learners_clust.kmodes`, `mlr_learners_clust.kproto`,
`mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.movMF`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.protoclust`,
`mlr_learners_clust.skmeans`, `mlr_learners_clust.som`, `mlr_learners_clust.specc`, `mlr_learners_clust.stdbsc`
`mlr_learners_clust.tclust`, `mlr_learners_clust.xmeans`

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.diana")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.em *Expectation-Maximization Clustering Learner*

Description

Expectation-Maximization clustering. Calls the EM Weka clusterer from package **RWeka**.

The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data. The learner supports both partitional and fuzzy clustering.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.em")
lrn("clust.em")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3cluster**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
I	integer	100		$[1, \infty)$
ll_cv	numeric	1e-06		$[1e-06, \infty)$
ll_iter	numeric	1e-06		$[1e-06, \infty)$
M	numeric	1e-06		$[1e-06, \infty)$
max	integer	-1		$[-1, \infty)$
N	integer	-1		$[-1, \infty)$
num_slots	integer	1		$[1, \infty)$
S	integer	100		$[0, \infty)$
X	integer	10		$[1, \infty)$
K	integer	10		$[1, \infty)$
V	logical	FALSE	TRUE, FALSE	-
O	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustEM`

Active bindings

`marshaled (logical(1))`
Whether the learner's model is marshaled.

Methods

Public methods:

- `LearnerClustEM$new()`
- `LearnerClustEM$marshal()`
- `LearnerClustEM$unmarshal()`
- `LearnerClustEM$clone()`

`LearnerClustEM$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustEM$new()
```

`LearnerClustEM$marshal()`: Marshal the learner's model.

Usage:

```
LearnerClustEM$marshal(...)
```

Arguments:

... (any)
Additional arguments passed to `mlr3::marshal_model()`.

`LearnerClustEM$unmarshal()`: Unmarshal the learner's model.

Usage:

```
LearnerClustEM$unmarshal(...)
```

Arguments:

... (any)
Additional arguments passed to `mlr3::unmarshal_model()`.

`LearnerClustEM$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustEM$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

Witten, H I, Frank, Eibe (2002). "Data mining: practical machine learning tools and techniques with Java implementations." *Acm Sigmod Record*, **31**(1), 76–77.

Dempster, P A, Laird, M N, Rubin, B D (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society: series B (methodological)*, **39**(1), 1–22.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.diana`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featureless`,
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,
`mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kcca`, `mlr_learners_clust.kkme`

mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbs mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.em")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.fanny

Fuzzy Analysis Clustering Learner

Description

Fuzzy Analysis (FANNY) clustering. Calls `cluster::fanny()` from package **cluster**.

The k parameter is set to 2 by default since `cluster::fanny()` doesn't have a default value for the number of clusters. The predict method copies cluster assignments and memberships generated for train data. The predict does not work for new data.

Initial parameter values

- `keep.diss`:
  - Actual default: n < 100, where n is the number of observations.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the dissimilarity matrix in the model to save memory.
- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.fanny")
lrn("clust.fanny")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
memb.exp	numeric	2		$[1, \infty)$
metric	character	euclidean	euclidean, manhattan, SqEuclidean	-
stand	logical	FALSE	TRUE, FALSE	-
iniMem.p	untyped	NULL		-
keep.diss	logical	-	TRUE, FALSE	-
keep.data	logical	TRUE	TRUE, FALSE	-
maxit	integer	500		$[0, \infty)$
tol	numeric	1e-15		$[0, \infty)$
trace.lev	integer	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustFanny`

Methods

Public methods:

- `LearnerClustFanny$new()`
- `LearnerClustFanny$clone()`

`LearnerClustFanny$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustFanny$new()
```

`LearnerClustFanny$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustFanny$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

Kaufman, Leonard, Rousseeuw, J P (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.featureless, mlr_learners_clust.ff,
mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hcl
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.fanny")
print(learner)

# Define a Task
task = tsk("usarrests")
```

```
# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.featureless`
Featureless Clustering Learner

Description

Featureless clustering. Randomly (but evenly) assigns observations to `num_clusters` partitions (default: 1 partition).

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.featureless")
lrn("clust.featureless")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered", "POSIXct", "Date"
- Required Packages: **mlr3**, **mlr3cluster**

Parameters

Id	Type	Default	Range
num_clusters	integer	-	$[1, \infty)$

Super classes

mlr3::Learner -> LearnerClust -> LearnerClustFeatureless

Methods

Public methods:

- LearnerClustFeatureless$new()
- LearnerClustFeatureless$clone()

LearnerClustFeatureless$new(): Creates a new instance of this R6 class.

Usage:

```
LearnerClustFeatureless$new()
```

LearnerClustFeatureless$clone(): The objects of this class are cloneable with this method.

Usage:

```
LearnerClustFeatureless$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- as.data.table(mlr_learners) for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.ff,
mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hcl
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.featureless")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.ff *Farthest First Clustering Learner*

Description

Farthest First clustering. Calls `RWeka::FarthestFirst()` from package **RWeka**.

The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.ff")
lrn("clust.ff")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3cluster**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
N	integer	2		$[1, \infty)$
S	integer	1		$[0, \infty)$
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustFarthestFirst`

Active bindings

`marshaled (logical(1))`
Whether the learner's model is marshaled.

Methods

Public methods:

- `LearnerClustFarthestFirst$new()`
- `LearnerClustFarthestFirst$marshal()`
- `LearnerClustFarthestFirst$unmarshal()`
- `LearnerClustFarthestFirst$clone()`

`LearnerClustFarthestFirst$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustFarthestFirst$new()
```

`LearnerClustFarthestFirst$marshal()`: Marshal the learner's model.

Usage:

```
LearnerClustFarthestFirst$marshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::marshal_model()`.

`LearnerClustFarthestFirst$unmarshal()`: Unmarshal the learner's model.

Usage:

```
LearnerClustFarthestFirst$unmarshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::unmarshal_model()`.

`LearnerClustFarthestFirst$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustFarthestFirst$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

Witten, H I, Frank, Eibe (2002). "Data mining: practical machine learning tools and techniques with Java implementations." *Acm Sigmod Record*, **31**(1), 76–77.

Hochbaum, S D, Shmoys, B D (1985). "A best possible heuristic for the k-center problem." *Mathematics of operations research*, **10**(2), 180–184.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hcl
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.ff")
print(learner)
```

```
# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.flexmix`

Finite Mixture Model Clustering Learner

Description

Finite mixture model clustering via the EM algorithm. Calls `flexmix::stepFlexmix()` from package **flexmix**, which runs `nrep` EM repetitions and keeps the best fit. When `cluster` provides fixed initial assignments, `flexmix::flexmix()` is called instead and `nrep` must not be larger than 1.

The k parameter is set to 2 by default since `flexmix::stepFlexmix()` doesn't have a default value for the number of components. Unset k, e.g. via `learner$param_set$set_values(k = NULL)`, to infer the number of components from `cluster` instead. The component model is selected through the `model` parameter, exposing the multivariate normal, univariate normal, multivariate binary, and multivariate Poisson drivers shipped with flexmix. The predict method calls `flexmix::clusters()` for cluster assignments and `flexmix::posterior()` for component probabilities on new data.

Note that EM can prune components whose prior falls below `minprior` during fitting, so the final number of components may be smaller than k.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.flexmix")
lrn("clust.flexmix")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **flexmix**, **mvtnorm**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
model	character	-	FLXMCmvnorm, FLXMCnorm1, FLXMCmvbinary, FLXMCmvpois	-
diagonal	logical	TRUE	TRUE, FALSE	-
truncated	logical	FALSE	TRUE, FALSE	-
cluster	untyped	NULL		-
iter.max	integer	200		$[1, \infty)$
minprior	numeric	0.05		$[0, 1]$
tolerance	numeric	1e-06		$[0, \infty)$
verbose	integer	0		$[0, \infty)$
classify	character	auto	auto, weighted, CEM, SEM, hard, random	-
nrep	integer	3		$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustFlexmix`

Methods

Public methods:

- `LearnerClustFlexmix$new()`
- `LearnerClustFlexmix$clone()`

`LearnerClustFlexmix$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustFlexmix$new()
```

`LearnerClustFlexmix$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustFlexmix$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Leisch, Friedrich (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18. doi:10.18637/jss.v011.i08.

Grün, Bettina, Leisch, Friedrich (2008). "FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. doi:10.18637/jss.v028.i04.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hclust,
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.flexmix")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.genie`

Genie Hierarchical Clustering Learner

Description

Genie hierarchical clustering, a fast and robust outlier-resistant algorithm based on the Gini inequality measure applied to cluster sizes during the linkage process. Calls `genieclust::gclust()` from package **genieclust**.

There is no predict method for `genieclust::gclust()`, so the method returns cluster labels for the training data obtained via `stats::cutree()` at the requested `k`. The task must have at least 2 features.

Custom mlr3 parameters

- k:
  - Not an argument of `genieclust::gclust()`. The number of clusters to cut the tree into, passed to `stats::cutree()`. Initialized to 2.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.genie")
lrn("clust.genie")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **genieclust**

Parameters

Id	Type	Default	Levels	Range
gini_threshold	numeric	0.3		$[0, 1]$
M	integer	0		$[0, \infty)$
distance	character	euclidean	euclidean, l2, manhattan, cityblock, l1, cosine	-
verbose	logical	FALSE	TRUE, FALSE	-
k	integer	-		$[1, \infty)$

Super classes

mlr3::Learner -> LearnerClust -> LearnerClustGenie

Methods

Public methods:

- LearnerClustGenie$new()
- LearnerClustGenie$clone()

`LearnerClustGenie$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustGenie$new()
```

`LearnerClustGenie$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustGenie$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Gagolewski, Marek, Bartoszuk, Maciej, Cena, Anna (2016). "Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm." *Information Sciences*, **363**, 8–23. doi:10.1016/j.ins.2016.05.003.

Gagolewski, Marek (2021). "genieclust: Fast and robust hierarchical clustering." *SoftwareX*, **15**, 100722. doi:10.1016/j.softx.2021.100722.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs

mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.gmeans, mlr_learners_clust.hclust
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.genie")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.gmeans`

G-Means Clustering Learner

Description

G-means clustering. Calls `gmeans::gmeans()` from package **gmeans**.

G-means extends k-means by automatically determining the number of clusters: starting from `k_init` centers, each cluster is repeatedly split in two unless an Anderson-Darling test suggests its points already follow a Gaussian distribution, until no more centers are added or `k_max` is reached. The predict method assigns new observations to the nearest cluster center.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.gmeans")
lrn("clust.gmeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **gmeans**

Parameters

Id	Type	Default	Levels	Range
k_init	integer	2		$[1, \infty)$
k_max	integer	10		$[1, \infty)$
level	numeric	0.05		$[0, 1]$
iter.max	integer	10		$[1, \infty)$
algorithm	character	Hartigan-Wong	Hartigan-Wong, Lloyd, Forgy, MacQueen	-
trace	logical	FALSE	TRUE, FALSE	-
method	character	euclidean	euclidean, manhattan, minkowski	-
p	numeric	2		$[0, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustGMeans`

Methods

Public methods:

- `LearnerClustGMeans$new()`
- `LearnerClustGMeans$clone()`

`LearnerClustGMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustGMeans$new()
```

`LearnerClustGMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustGMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Hamerly, Greg, Elkan, Charles (2003). "Learning the k in k-means." In *Advances in Neural Information Processing Systems*, volume 16.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.hclust,
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.gmeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.hclust` *Hierarchical Clustering Learner*

Description

Agglomerative hierarchical clustering. Calls `stats::hclust()` from package **stats**.

Distance calculation is done by `stats::dist()`.

Custom mlr3 parameters

- `distmethod`:
  - Not an argument of `stats::hclust()`. The distance measure, passed as `method` to `stats::dist()` when computing the dissimilarity matrix.
- `diag`, `upper`, `p`:
  - Not arguments of `stats::hclust()`. Forwarded to `stats::dist()` when computing the dissimilarity matrix.
- `k`:
  - Not an argument of `stats::hclust()`. The number of clusters to cut the tree into, passed to `stats::cutree()`. Initialized to `2`.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.hclust")
lrn("clust.hclust")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3cluster**, ’stats’

Parameters

Id	Type	Default	Levels	Range
method	character	complete	ward.D, ward.D2, single, complete, average, mcquitty, median, centroid	-
members	untyped	NULL		-
distmethod	character	euclidean	euclidean, maximum, manhattan, canberra, binary, minkowski	-
diag	logical	FALSE	TRUE, FALSE	-

upper	logical	FALSE	TRUE, FALSE	-
p	numeric	2		$[0, \infty)$
k	integer	-		$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustHclust`

Methods

Public methods:

- `LearnerClustHclust$new()`
- `LearnerClustHclust$clone()`

`LearnerClustHclust$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustHclust$new()
```

`LearnerClustHclust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustHclust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Becker, A R, Chambers, M J, Wilks, R A (1988). *The New S Language*. Wadsworth & Brooks/Cole.

Everitt, S B (1974). *Cluster Analysis*. Heinemann Educational Books.

Hartigan, A J (1975). *Clustering Algorithms*. John Wiley & Sons.

Sneath, HA P, Sokal, R R (1973). *Numerical Taxonomy*. Freeman.

Anderberg, R M (1973). *Cluster Analysis for Applications*. Academic Press.

Gordon, David A (1999). *Classification*, 2 edition. Chapman and Hall / CRC.

Murtagh, Fionn (1985). "Multidimensional Clustering Algorithms." In *COMPSTAT Lectures 4*. Physica-Verlag.

McQuitty, L L (1966). "Similarity Analysis by Reciprocal Pairs for Discrete and Continuous Data." *Educational and Psychological Measurement*, **26**(4), 825–831. doi:10.1177/001316446602600402.

Legendre, Pierre, Legendre, Louis (2012). *Numerical Ecology*, 3 edition. Elsevier Science BV.

Murtagh, Fionn, Legendre, Pierre (2014). "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?" *Journal of Classification*, **31**, 274–295. doi:10.1007/s003570149161z.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.hclust")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.hdbscan`
HDBSCAN Clustering Learner

Description

HDBSCAN (hierarchical DBSCAN) clustering. Calls `dbscan::hdbscan()` from package **dbscan**.

The `minPts` parameter is set to 5 by default since `dbscan::hdbscan()` doesn't have a default value for the minimum size of clusters.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.hdbscan")
lrn("clust.hdbscan")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **dbscan**

Parameters

Id	Type	Default	Levels	Range
minPts	integer	-		$[2, \infty)$
cluster_selection_epsilon	numeric	0		$(-\infty, \infty)$
gen_hdbscan_tree	logical	FALSE	TRUE, FALSE	-
gen_simplified_tree	logical	FALSE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustHDBSCAN`

Active bindings

`native_model` (hdbscan)
The fitted model returned by `dbscan::hdbscan()`.

Methods

Public methods:

- `LearnerClustHDBSCAN$new()`
- `LearnerClustHDBSCAN$clone()`

`LearnerClustHDBSCAN$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustHDBSCAN$new()
```

`LearnerClustHDBSCAN$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustHDBSCAN$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Hahsler M, Piekenbrock M, Doran D (2019). "dbscan: Fast Density-Based Clustering with R." *Journal of Statistical Software*, **91**(1), 1–30. doi:10.18637/jss.v091.i01.

Campello, JGB R, Moulavi, Davoud, Sander, Jörg (2013). "Density-based clustering based on hierarchical density estimates." In *Pacific-Asia conference on knowledge discovery and data mining*, 160–172. Springer.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,

mlr_learners_clust.hclust, mlr_learners_clust.kcca, mlr_learners_clust.kkmeans, mlr_learners_clust.kmea
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbs
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.hdbscan")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.kcca

K-Centroids Cluster Analysis Learner

Description

K-Centroids Cluster Analysis - a unified framework for partitional clustering with selectable distance / centroid families: standard k-means, k-medians, spherical k-means (`"angle"`), Jaccard, and extended Jaccard. Calls `flexclust::kcca()` from package **flexclust**.

The k parameter is set to 2 by default since `flexclust::kcca()` has no default value for the number of clusters. The predict method uses `flexclust::clusters()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kcca")
lrn("clust.kcca")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **flexclust**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[2, \infty)$
family	character	kmeans	kmeans, kmedians, angle, jaccard, ejaccard	-
weights	untyped	NULL		-
group	untyped	NULL		-
simple	logical	FALSE	TRUE, FALSE	-
save.data	logical	FALSE	TRUE, FALSE	-
iter.max	integer	200		$[1, \infty)$
tolerance	numeric	1e-06		$[0, \infty)$
verbose	integer	0		$[0, \infty)$
classify	character	auto	auto, weighted, hard, simann	-
initcent	untyped	"randomcent"		-
gamma	numeric	1		$[0, \infty)$
simann	untyped	c(0.3, 0.95, 10)		-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKCCA`

Methods

Public methods:

- `LearnerClustKCCA$new()`
- `LearnerClustKCCA$clone()`

`LearnerClustKCCA$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKCCA$new()
```

`LearnerClustKCCA$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKCCA$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Leisch, Friedrich (2006). "A Toolbox for K-Centroids Cluster Analysis." *Computational Statistics & Data Analysis*, **51**(2), 526–544. doi:10.1016/j.csda.2005.10.006.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmeans,
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.kcca")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
```

```
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.kkmeans`

Kernel K-Means Clustering Learner

Description

Kernel k-means clustering. Calls `kernlab::kkmeans()` from package **kernlab**.

The `centers` parameter is set to 2 by default since `kernlab::kkmeans()` doesn't have a default value for the number of clusters. The predict method assigns each new observation to the cluster whose centroid is nearest in the kernel-induced feature space, computed from the stored training data. The model is therefore a list containing the fitted `kernlab::kkmeans()` object along with the training data and per-cluster kernel statistics. The task must have at least 2 features.

Custom mlr3 parameters

- `sigma`, `degree`, `scale`, `offset`, `order`:
  - In `kernlab::kkmeans()`, these kernel parameters are passed inside the `kpar` list. Here they are exposed directly and assembled into `kpar` internally.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kkmeans")
lrn("clust.kkmeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **kernlab**

Parameters

Id	Type	Default	Levels	Range
centers	untyped	-		-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		$[0, \infty)$
degree	integer	1		$[1, \infty)$
scale	numeric	1		$[0, \infty)$
offset	numeric	1		$(-\infty, \infty)$
order	integer	1		$(-\infty, \infty)$
alg	character	kkmeans	kkmeans, kerninghan	-
p	numeric	1		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKKMeans`

Active bindings

`native_model (specc)`
The fitted model returned by `kernlab::kkmeans()`.

Methods

Public methods:

- `LearnerClustKKMeans$new()`
- `LearnerClustKKMeans$clone()`

`LearnerClustKKMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKKMeans$new()
```

`LearnerClustKKMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKKMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Karatzoglou, Alexandros, Smola, Alexandros, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**, 1–20.

Dhillon, S I, Guan, Yuqiang, Kulis, Brian (2004). *A unified view of kernel k-means, spectral clustering and graph cuts*. Citeseer.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kmea
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.kkmeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.kmeans`
K-Means Clustering Learner

Description

K-means clustering. Calls `stats::kmeans()` from package **stats**.

The `centers` parameter is set to 2 by default since `stats::kmeans()` doesn't have a default value for the number of clusters. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kmeans")
lrn("clust.kmeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, 'stats', **clue**

Parameters

Id	Type	Default	Levels	Range
centers	untyped	-		-
iter.max	integer	10		$[1, \infty)$
algorithm	character	Hartigan-Wong	Hartigan-Wong, Lloyd, Forgy, MacQueen	-
nstart	integer	1		$[1, \infty)$
trace	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKMeans`

Methods

Public methods:

- `LearnerClustKMeans$new()`
- `LearnerClustKMeans$clone()`

`LearnerClustKMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKMeans$new()
```

`LearnerClustKMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Forgy, W E (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications." *Biometrics*, **21**, 768–769.

Hartigan, A J, Wong, A M (1979). "Algorithm AS 136: A K-means clustering algorithm." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **28**(1), 100–108. doi:10.2307/2346830.

Lloyd, P S (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**(2), 129–137.

MacQueen, James (1967). "Some methods for classification and analysis of multivariate observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, 281–297.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners`
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.

- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.kmeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.kmeans_rcpp`

K-Means Clustering Learner from ClusterR

Description

K-means clustering. Calls `ClusterR::KMeans_rcpp()` from package **ClusterR**.

The `clusters` parameter is set to 2 by default since `ClusterR::KMeans_rcpp()` doesn't have a default value for the number of clusters. The predict method computes the cluster memberships for new data via the fitted centroids.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kmeans_rcpp")
lrn("clust.kmeans_rcpp")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **ClusterR**

Parameters

Id	Type	Default	Levels	Range
clusters	integer	-		$[1, \infty)$
num_init	integer	1		$[1, \infty)$
max_iters	integer	100		$[1, \infty)$
initializer	character	kmeans++	optimal_init, quantile_init, kmeans++, random	-
verbose	logical	FALSE	TRUE, FALSE	-
CENTROIDS	untyped	NULL		-
tol	numeric	1e-04		$[0, \infty)$
tol_optimal_init	numeric	0.3		$[0, \infty)$
seed	integer	1		$(-\infty, \infty)$
threads	integer	1		$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKMeansRcpp`

Methods

Public methods:

- `LearnerClustKMeansRcpp$new()`
- `LearnerClustKMeansRcpp$clone()`

`LearnerClustKMeansRcpp$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKMeansRcpp$new()
```

`LearnerClustKMeansRcpp$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKMeansRcpp$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

Hartigan, A J, Wong, A M (1979). "Algorithm AS 136: A K-means clustering algorithm." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **28**(1), 100–108. doi:10.2307/2346830.

Lloyd, P S (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**(2), 129–137.

Arthur, David, Vassilvitskii, Sergei (2007). "k-means++: the advantages of careful seeding." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmodes, mlr_learners_clust.kproto, mlr_learners_clust.mcl
mlr_learners_clust.meanshift, mlr_learners_clust.movMF, mlr_learners_clust.optics,
mlr_learners_clust.pam, mlr_learners_clust.protoclust, mlr_learners_clust.skmeans,
mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbscan, mlr_learners_clust.tclus
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.kmeans_rcpp")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.kmodes`

K-Modes Clustering Learner

Description

K-modes clustering for categorical data. Calls `klaR::kmodes()` from package **klaR**.

All feature values are treated as categories. Numeric features are accepted, but `klaR::kmodes()` warns when they contain more than 30 distinct values. Logical features are converted to factors with levels `FALSE` and `TRUE`.

The `modes` parameter is set to 2 by default since `klaR::kmodes()` does not have a default value for the number of clusters. Since `klaR::kmodes()` does not provide a predict method, new observations are assigned to their closest learned mode. Prediction always uses unweighted simple matching distance, including for models trained with `weighted = TRUE`.

Custom mlr3 parameters

- `ties`:
  - Not an argument of `klaR::kmodes()`. Controls how ties are broken when assigning observations to their closest mode during prediction.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kmodes")
lrn("clust.kmodes")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3cluster**, **klaR**

Parameters

Id	Type	Default	Levels	Range
modes	untyped	-		-
iter.max	integer	10		$[1, \infty)$
weighted	logical	FALSE	TRUE, FALSE	-
fast	logical	TRUE	TRUE, FALSE	-
ties	character	first	first, last, random	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKModes`

Methods

Public methods:

- `LearnerClustKModes$new()`
- `LearnerClustKModes$clone()`

`LearnerClustKModes$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKModes$new()
```

`LearnerClustKModes$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKModes$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Huang, Zhexue (1997). “A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining.” In *Data Mining: Techniques and Applications*, 21–34.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kproto,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the learner
learner = lrn("clust.kmodes", modes = 2L)

# Define a categorical task
data = data.frame(
  color = factor(c("red", "red", "blue", "blue")),
  shape = factor(c("round", "round", "square", "square"))
)
task = as_task_clust(data)

# Train and predict
learner$train(task)
prediction = learner$predict(task)
```

`mlr_learners_clust.kproto`

K-Prototypes Clustering Learner

Description

K-prototypes clustering for mixed-type data. Calls `clustMixType::kproto()` from package **clustMixType**.

The `k` parameter is set to 2 by default since `clustMixType::kproto()` doesn't have a default value for the number of clusters. The task must have at least 2 features. Logical features are converted to factors with levels `FALSE` and `TRUE`, so they enter the categorical part of the distance. Thus, a vector-valued `lambda` needs one entry per numeric, logical, and factor feature.

Initial parameter values

- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.
- `verbose`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Suppress verbose output during training.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.kproto")
lrn("clust.kproto")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3cluster**, **clustMixType**

Parameters

Id	Type	Default	Levels	Range
k	untyped	-		-
lambda	untyped	NULL		-
type	character	huang	huang, gower	-
iter.max	integer	100		$[1, \infty)$
nstart	integer	1		$[1, \infty)$
na.rm	character	yes	yes, no, imp.internal, imp.onestep	-
keep.data	logical	TRUE	TRUE, FALSE	-
verbose	logical	TRUE	TRUE, FALSE	-
init	character	NULL	nbh.dens, sel.cen, nstart.m	-
p_nstart.m	numeric	0.9		$[0, 1]$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustKProto`

Methods

Public methods:

- `LearnerClustKProto$new()`
- `LearnerClustKProto$clone()`

`LearnerClustKProto$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustKProto$new()
```

`LearnerClustKProto$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustKProto$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

Huang, Zhexue (1998). "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values." *Data Mining and Knowledge Discovery*, **2**(3), 283–304.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.kproto")
print(learner)

# Define a mixed-type Task (kproto requires at least one factor variable)
data = data.frame(
  x1 = c(1, 2, 10, 11, 1, 2, 10, 11),
  x2 = factor(c("a", "a", "b", "b", "a", "a", "b", "b"))
)
task = as_task_clust(data)

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$model)

# Make predictions for the task
prediction = learner$predict(task)
```

```
# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.MBatchKMeans` *Mini Batch K-Means Clustering Learner*

Description

Mini-batch k-means clustering. Calls `ClusterR::MiniBatchKmeans()` from package **ClusterR**.

The `clusters` parameter is set to 2 by default since `ClusterR::MiniBatchKmeans()` doesn't have a default value for the number of clusters. The predict method uses `ClusterR::predict_KMeans()` on the fitted centroids to compute the cluster memberships for new data. The learner supports both partitional and fuzzy clustering.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.MBatchKMeans")
lrn("clust.MBatchKMeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **ClusterR**

Parameters

Id	Type	Default	Levels	Range
clusters	integer	-		$[1, \infty)$
batch_size	integer	10		$[1, \infty)$
num_init	integer	1		$[1, \infty)$
max_iters	integer	100		$[1, \infty)$
init_fraction	numeric	1		$[0, 1]$
initializer	character	kmeans++	optimal_init, quantile_init, kmeans++, random	-
early_stop_iter	integer	10		$[1, \infty)$
verbose	logical	FALSE	TRUE, FALSE	-
CENTROIDS	untyped	NULL		-
tol	numeric	1e-04		$[0, \infty)$

tol_optimal_init	numeric	0.3	$[0, \infty)$
seed	integer	1	$(-\infty, \infty)$
threads	integer	1	$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustMiniBatchKMeans`

Methods

Public methods:

- `LearnerClustMiniBatchKMeans$new()`
- `LearnerClustMiniBatchKMeans$clone()`

`LearnerClustMiniBatchKMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustMiniBatchKMeans$new()
```

`LearnerClustMiniBatchKMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustMiniBatchKMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Sculley, David (2010). "Web-scale k-means clustering." In *Proceedings of the 19th international conference on World wide web*, 1177–1178.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners`
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.

  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agnes, mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara, mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc, mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.MBatchKMeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.mclust

Gaussian Mixture Model Clustering Learner

Description

Gaussian mixture model-based clustering. Calls mclust::Mclust() from package **mclust**.

The predict method uses mclust::predict.Mclust() to compute the cluster memberships for new data.

Initial parameter values

- `verbose`:
  - Actual default: `interactive()`.
  - Adjusted default: `FALSE`.
  - Reason for change: Suppress progress output during training.
- `warn`:
  - Actual default: `mclust.options("warn")`, which is `FALSE` by default.
  - Adjusted default: `FALSE`.
  - Reason for change: Suppress warnings during training independently of the `mclust` global options.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.mclust")
lrn("clust.mclust")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition", "prob"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **mclust**

Parameters

Id	Type	Default	Levels
G	untyped	1:9	
modelNames	untyped	NULL	
prior	untyped	NULL	
control	untyped	-	
initialization	untyped	NULL	
warn	logical	-	TRUE, FALSE
x	untyped	NULL	
verbose	logical	-	TRUE, FALSE

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustMclust`

Methods

Public methods:

- `LearnerClustMclust$new()`
- `LearnerClustMclust$clone()`

`LearnerClustMclust$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustMclust$new()
```

`LearnerClustMclust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustMclust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Scrucca, Luca, Fop, Michael, Murphy, Brendan T, Raftery, E A (2016). "mclust 5: clustering, classification and density estimation using Gaussian finite mixture models." *The R journal*, **8**(1), 289.

Fraley, Chris, Raftery, E A (2002). "Model-based clustering, discriminant analysis, and density estimation." *Journal of the American statistical Association*, **97**(458), 611–631.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,

mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.meanshift, mlr_learners_clust.movMF,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.mclust")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

---

mlr_learners_clust.meanshift

Mean Shift Clustering Learner

Description

Mean shift clustering. Calls LPCM::ms() from package **LPCM**.

The predict method runs the mean-shift iteration from each new observation with the trained bandwidth via LPCM::ms.rep() and assigns it to the mode it converges to, i.e. the basin of attraction of the fitted modes.

Initial parameter values

- plot:
  - Actual default: TRUE.
  - Adjusted default: FALSE.
  - Reason for change: Suppress plotting during training.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.meanshift")
lrn("clust.meanshift")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **LPCM**

Parameters

Id	Type	Default	Levels	Range
h	untyped	-		-
subset	untyped	-		-
thr	numeric	0.01		$(-\infty, \infty)$
scaled	integer	1		$[0, \infty)$
iter	integer	200		$[1, \infty)$
plot	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustMeanShift`

Active bindings

`native_model (ms)`
The fitted model returned by `LPCM::ms()`.

Methods

Public methods:

- `LearnerClustMeanShift$new()`
- `LearnerClustMeanShift$clone()`

`LearnerClustMeanShift$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustMeanShift$new()
```

`LearnerClustMeanShift$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustMeanShift$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Cheng, Yizong (1995). "Mean shift, mode seeking, and clustering." *IEEE transactions on pattern analysis and machine intelligence*, **17**(8), 790–799.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.movMF, mlr_learners_clust.opti
mlr_learners_clust.pam, mlr_learners_clust.protoclust, mlr_learners_clust.skmeans,
mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbscan, mlr_learners_clust.tclus
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.meanshift")
print(learner)

# Define a Task
task = tsk("usarrests")
```

```
# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.movMF`

von Mises-Fisher Mixture Clustering Learner

Description

Fits a mixture of von Mises-Fisher distributions via EM, the directional-data analogue of a Gaussian mixture for points on the unit hypersphere. Calls `movMF::movMF()` from package **movMF**.

The `k` parameter is set to 2 by default since `movMF::movMF()` has no default value for the number of mixture components. Rows of `x` are standardized to unit length internally by `movMF::movMF()`. Predictions use the `predict()` method from **movMF**; `prob` returns the soft memberships.

Setting `ids` initializes the EM algorithm from fixed component memberships and stops it after a single iteration, in which case `movMF::movMF()` ignores `start` and `nruns`.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.movMF")
lrn("clust.movMF")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3cluster**, **movMF**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
E	character	softmax	softmax, hardmax, stochmax	-
kappa	untyped	-		-
ids	untyped	-		-
start	untyped	"p"		-
nruns	integer	1		$[1, \infty)$
maxiter	integer	100		$[1, \infty)$
reltol	numeric	-		$[0, \infty)$
minalpha	numeric	0		$[0, \infty)$
converge	logical	-	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustMovMF`

Methods

Public methods:

- `LearnerClustMovMF$new()`
- `LearnerClustMovMF$clone()`

`LearnerClustMovMF$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustMovMF$new()
```

`LearnerClustMovMF$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustMovMF$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Banerjee, Arindam, Dhillon, S I, Ghosh, Joydeep, Sra, Suvrit (2005). "Clustering on the Unit Hypersphere using von Mises-Fisher Distributions." *Journal of Machine Learning Research*, **6**(46), 1345–1382.

Hornik, Kurt, Grün, Bettina (2014). "movMF: An R Package for Fitting Mixtures of von Mises-Fisher Distributions." *Journal of Statistical Software*, **58**(10), 1–31. doi:10.18637/jss.v058.i10.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.movMF")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.optics`
OPTICS Clustering Learner

Description

OPTICS (ordering points to identify the clustering structure) clustering. Calls `dbscan::optics()` from package **dbscan**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.optics")
lrn("clust.optics")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **dbscan**

Parameters

Id	Type	Default	Levels	Range
eps	numeric	Inf		$[0, \infty)$
minPts	integer	5		$[1, \infty)$
search	character	kdtree	kdtree, linear, dist	-
bucketSize	integer	10		$[1, \infty)$
splitRule	character	SUGGEST	STD, MIDPT, FAIR, SL_MIDPT, SL_FAIR, SUGGEST	-
approx	numeric	0		$[0, \infty)$
eps_cl	numeric	-		$[0, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustOPTICS`

Active bindings

`native_model` (`optics`)
The fitted model returned by `dbscan::optics()`, with the clustering extracted by `dbscan::extractDBSCAN()`.

Methods

Public methods:

- `LearnerClustOPTICS$new()`
- `LearnerClustOPTICS$clone()`

`LearnerClustOPTICS$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustOPTICS$new()
```

`LearnerClustOPTICS$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustOPTICS$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Hahsler M, Piekenbrock M, Doran D (2019). "dbscan: Fast Density-Based Clustering with R." *Journal of Statistical Software*, **91**(1), 1–30. doi:10.18637/jss.v091.i01.

Ankerst, Mihael, Breunig, M M, Kriegel, Hans-Peter, Sander, Jörg (1999). "OPTICS: Ordering points to identify the clustering structure." *ACM Sigmod record*, **28**(2), 49–60.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners`
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,
`mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kcca`, `mlr_learners_clust.kkme`

mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF, mlr_learners_clust.pam, mlr_learners_clust.protoclust, mlr_learners_clust.skm mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbscan, mlr_learners_clust.tclus mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.optics")
print(learner)
```

`mlr_learners_clust.pam`

Partitioning Around Medoids Clustering Learner

Description

Partitioning Around Medoids (PAM) clustering. Calls `cluster::pam()` from package **cluster**.

The `k` parameter is set to 2 by default since `cluster::pam()` doesn't have a default value for the number of clusters. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Initial parameter values

- `keep.diss`:
  - Actual default: `n < 100`, where `n` is the number of observations.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the dissimilarity matrix in the model to save memory.
- `keep.data`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.pam")
lrn("clust.pam")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**, **clue**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
metric	character	euclidean	euclidean, manhattan	-
medoids	untyped	NULL		-
nstart	integer	-		$[1, \infty)$
stand	logical	FALSE	TRUE, FALSE	-
do.swap	logical	TRUE	TRUE, FALSE	-
keep.diss	logical	-	TRUE, FALSE	-
keep.data	logical	TRUE	TRUE, FALSE	-
pamonce	untyped	FALSE		-
variant	character	original	original, o_1, o_2, f_3, f_4, f_5, faster	-
trace.lev	integer	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustPAM`

Methods

Public methods:

- `LearnerClustPAM$new()`
- `LearnerClustPAM$clone()`

`LearnerClustPAM$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustPAM$new()
```

`LearnerClustPAM$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustPAM$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Reynolds, P A, Richards, Graeme, de la Iglesia, Beatriz, Rayward-Smith, J V (2006). "Clustering rules: a comparison of partitioning and hierarchical clustering algorithms." *Journal of Mathematical Modelling and Algorithms*, **5**, 475–504.

Schubert, Erich, Rousseeuw, J P (2019). "Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms." In *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings 12*, 171–187. Springer.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.protoclust,
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.pam")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)
```

```
# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.protoclust`
Prototype Hierarchical Clustering Learner

Description

Hierarchical clustering using minimax linkage with prototypes. Calls `protoclust::protoclust()` from package **protoclust**.

The predict method cuts the tree at the current `k` via `protoclust::protocut()` and assigns each new observation to the cluster of its nearest prototype, using the same distance method as during training. The model is therefore a list containing the fitted `protoclust::protoclust()` object, along with the training data and the arguments passed to `stats::dist()`.

Custom mlr3 parameters

- k:
  - Not an argument of `protoclust::protoclust()`. The number of clusters to cut the tree into, passed to `protoclust::protocut()`. Initialized to 2.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.protoclust")
lrn("clust.protoclust")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **protoclust**

Parameters

Id	Type	Default	Levels	Range
method	character	euclidean	euclidean, maximum, manhattan, canberra, binary, minkowski	-
diag	logical	FALSE	TRUE, FALSE	-
upper	logical	FALSE	TRUE, FALSE	-
p	numeric	2		$[0, \infty)$
verb	logical	FALSE	TRUE, FALSE	-
k	integer	-		$[1, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustProtoclust`

Active bindings

`native_model` (`protoclust`)
The fitted model returned by `protoclust::protoclust()`.

Methods

Public methods:

- `LearnerClustProtoclust$new()`
- `LearnerClustProtoclust$clone()`

`LearnerClustProtoclust$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustProtoclust$new()
```

`LearnerClustProtoclust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustProtoclust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Bien, Jacob, Tibshirani, Robert (2011). "Hierarchical Clustering with Prototypes via Minimax Linkage." *Journal of the American Statistical Association*, **106**(495), 1075–1084.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.skmeans
mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbscan, mlr_learners_clust.tclus
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.protoclust")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.SimpleKMeans`
K-Means Clustering Learner (Weka)

Description

K-means clustering (Weka). Calls `RWeka::SimpleKMeans()` from package **RWeka**.

The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.SimpleKMeans")
lrn("clust.SimpleKMeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3cluster**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
A	untyped	"weka.core.EuclideanDistance"		-
C	logical	FALSE	TRUE, FALSE	-
fast	logical	FALSE	TRUE, FALSE	-
I	integer	500		$[1, \infty)$
init	integer	0		$[0, 3]$
M	logical	FALSE	TRUE, FALSE	-
max_candidates	integer	100		$[1, \infty)$
min_density	numeric	2		$[0, \infty)$
N	integer	2		$[1, \infty)$
num_slots	integer	1		$[1, \infty)$
O	logical	FALSE	TRUE, FALSE	-
periodic_pruning	integer	10000		$[1, \infty)$
S	integer	10		$[0, \infty)$
t2	numeric	-1		$(-\infty, \infty)$
t1	numeric	-1.25		$(-\infty, \infty)$
V	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-

do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustSimpleKMeans`

Active bindings

`marshaled (logical(1))`
Whether the learner's model is marshaled.

Methods

Public methods:

- `LearnerClustSimpleKMeans$new()`
- `LearnerClustSimpleKMeans$marshal()`
- `LearnerClustSimpleKMeans$unmarshal()`
- `LearnerClustSimpleKMeans$clone()`

`LearnerClustSimpleKMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustSimpleKMeans$new()
```

`LearnerClustSimpleKMeans$marshal()`: Marshal the learner's model.

Usage:

```
LearnerClustSimpleKMeans$marshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::marshal_model()`.

`LearnerClustSimpleKMeans$unmarshal()`: Unmarshal the learner's model.

Usage:

```
LearnerClustSimpleKMeans$unmarshal(...)
```

Arguments:

`...` (any)
Additional arguments passed to `mlr3::unmarshal_model()`.

`LearnerClustSimpleKMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustSimpleKMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Witten, H I, Frank, Eibe (2002). "Data mining: practical machine learning tools and techniques with Java implementations." *Acm Sigmod Record*, **31**(1), 76–77.

Forgy, W E (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications." *Biometrics*, **21**, 768–769.

Lloyd, P S (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**(2), 129–137.

MacQueen, James (1967). "Some methods for classification and analysis of multivariate observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, 281–297.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.agnes, mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara, mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_fpc, mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans, mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes, mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.SimpleKMeans")
print(learner)

# Define a Task
```

```
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.skmeans`

Spherical K-Means Clustering Learner

Description

Spherical k-means clustering for data on the unit hypersphere. Calls `skmeans::skmeans()` from package **skmeans**.

The `k` parameter is set to 2 by default since `skmeans::skmeans()` doesn't have a default value for the number of clusters. Observations are partitioned by maximizing cosine similarity to cluster prototypes. Predictions on new data assign each observation to the prototype with the highest cosine similarity. Rows with zero norm are not allowed by `skmeans::skmeans()`.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.skmeans")
lrn("clust.skmeans")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **skmeans**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[2, \infty)$
method	character	-	genetic, pclust, CLUTO, gmeans, LIH, LIHC	-
m	numeric	1		$[1, \infty)$
weights	untyped	1		-
start	untyped	"p"		-
maxiter	integer	-		$[1, \infty)$
nruns	integer	-		$[1, \infty)$
maxchains	integer	-		$[0, \infty)$
popsize	integer	-		$[1, \infty)$
mutations	numeric	-		$[0, 1]$
reltol	numeric	-		$[0, \infty)$
verbose	logical	-	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustSKMeans`

Methods

Public methods:

- `LearnerClustSKMeans$new()`
- `LearnerClustSKMeans$clone()`

`LearnerClustSKMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustSKMeans$new()
```

`LearnerClustSKMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustSKMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Dhillon, S I, Modha, S D (2001). "Concept decompositions for large sparse text data using clustering." *Machine Learning*, **42**(1), 143–175. doi:10.1023/A:1007612920971.

Hornik, Kurt, Feinerer, Ingo, Kober, Martin, Buchta, Christian (2012). "Spherical k-Means Clustering." *Journal of Statistical Software*, **50**(10), 1–22. doi:10.18637/jss.v050.i10.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbscan, mlr_learners_clust.tclus
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.skmeans")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.som`

Self-Organizing Maps Clustering Learner

Description

Self-organizing map (Kohonen network) clustering. Calls `kohonen::som()` from package **kohonen**.

Each map unit corresponds to a cluster, so the number of clusters is `xdim * ydim`. The predict method uses `kohonen::map()` to assign new data to the closest unit.

Custom mlr3 parameters

- `xdim`, `ydim`, `topo`, `neighbourhood.fct`, `toroidal`:
  - Not arguments of `kohonen::som()`. These construct the map grid and are forwarded to `kohonen::somgrid()`.

Initial parameter values

- `cores`:
  - Actual default: `-1L`, using all available cores.
  - Adjusted default: `1L`.
  - Reason for change: Conflicting with parallelization via **future**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.som")
lrn("clust.som")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3cluster**, **kohonen**

Parameters

Id	Type	Default	Levels	Range
xdim	integer	8		$[1, \infty)$
ydim	integer	6		$[1, \infty)$
topo	character	rectangular	rectangular, hexagonal	-
neighbourhood.fct	character	bubble	bubble, gaussian	-
toroidal	logical	FALSE	TRUE, FALSE	-
rlen	integer	100		$[1, \infty)$
alpha	untyped	c(0.05, 0.01)		-
radius	untyped	-		-
user.weights	untyped	1		-
maxNA.fraction	numeric	0		$[0, 1]$
keep.data	logical	TRUE	TRUE, FALSE	-
dist.fcts	untyped	NULL		-
mode	character	online	online, batch, pbatch	-
cores	integer	-1		$(-\infty, \infty)$
init	untyped	-		-
normalizeDataLayers	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustSOM`

Methods

Public methods:

- `LearnerClustSOM$new()`
- `LearnerClustSOM$clone()`

`LearnerClustSOM$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustSOM$new()
```

`LearnerClustSOM$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustSOM$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Kohonen, Teuvo (1990). "The self-organizing map." *Proceedings of the IEEE*, **78**(9), 1464–1480. doi:10.1109/5.58325.

Wehrens, Ron, Kruisselbrink, Johannes (2018). "Flexible self-organizing maps in kohonen 3.0." *Journal of Statistical Software*, **87**(7), 1–18. doi:10.18637/jss.v087.i07.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.specc, mlr_learners_clust.stdbscan,
mlr_learners_clust.tclust, mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.som")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.specc`

Spectral Clustering Learner

Description

Spectral clustering. Calls `kernlab::specc()` from package **kernlab**.

The `centers` parameter is set to 2 by default since `kernlab::specc()` doesn't have a default value for the number of clusters.

There is no predict method for `kernlab::specc()`, so the method returns cluster labels for the training data.

Custom mlr3 parameters

- `sigma`, `degree`, `scale`, `offset`, `order`:
  - In `kernlab::specc()`, these kernel parameters are passed inside the `kpar` list. Here they are exposed directly and assembled into `kpar` internally.
  - If `kernel` is `"rbfdot"` and `sigma` is not set, `sigma` is estimated from the data (`kpar = "automatic"`).

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.specc")
lrn("clust.specc")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **kernlab**

Parameters

Id	Type	Default	Levels	Ran
centers	integer	-		[2, ∘
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		[0, ∘
degree	integer	1		[1, ∘
scale	numeric	1		[0, ∘
offset	numeric	1		(−∘

order	integer	1		$(-\infty$
nystrom.red	logical	FALSE	TRUE, FALSE	-
nystrom.sample	integer	-		$[1, \infty$
iterations	integer	200		$[1, \infty$
mod.sample	numeric	0.75		$[0, 1$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustSpectral`

Methods

Public methods:

- `LearnerClustSpectral$new()`
- `LearnerClustSpectral$clone()`

`LearnerClustSpectral$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustSpectral$new()
```

`LearnerClustSpectral$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustSpectral$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Karatzoglou, Alexandros, Smola, Alexandros, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**, 1–20.

Ng, Y A, Jordan, I M, Weiss, Yair (2001). "On Spectral Clustering: Analysis and an Algorithm." In *Advances in Neural Information Processing Systems*, volume 14.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners`
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.

- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.stdbscan, mlr_learners_clust.tc
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.specc")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)

# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.stdbscan`

ST-DBSCAN Clustering Learner

Description

ST-DBSCAN (spatio-temporal density-based spatial clustering of applications with noise) clustering. Calls `stdbscan::st_dbscan()` from package **stdbscan**.

The task must have exactly 3 features: the first two features (in the task's feature order, which is alphabetical for newly created tasks) are used as the spatial coordinates and the third feature as the temporal coordinate.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.stdbscan")
lrn("clust.stdbscan")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **stdbscan**

Parameters

Id	Type	Default	Levels	Range
eps_spatial	numeric	-		$[0, \infty)$
eps_temporal	numeric	-		$[0, \infty)$
min_pts	integer	-		$[1, \infty)$
weights	untyped	NULL		-
borderPoints	logical	TRUE	TRUE, FALSE	-
search	character	kdtree	kdtree, linear, dist	-
bucketSize	integer	10		$[1, \infty)$
splitRule	character	SUGGEST	STD, MIDPT, FAIR, SL_MIDPT, SL_FAIR, SUGGEST	-
approx	numeric	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustSTDBSCAN`

Active bindings

`native_model` (stdbscan)
The fitted model returned by `stdbscan::st_dbscan()`.

Methods

Public methods:

- `LearnerClustSTDBSCAN$new()`
- `LearnerClustSTDBSCAN$clone()`

`LearnerClustSTDBSCAN$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustSTDBSCAN$new()
```

`LearnerClustSTDBSCAN$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustSTDBSCAN$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Birant, Derya, Kut, Alp (2007). "ST-DBSCAN: An algorithm for clustering spatial-temporal data." *Data & Knowledge Engineering*, **60**(1), 208–221. doi:10.1016/j.datak.2006.01.013.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: `mlr_learners_clust.MBatchKMeans`, `mlr_learners_clust.SimpleKMeans`, `mlr_learners_clust.agne`
`mlr_learners_clust.ap`, `mlr_learners_clust.bico`, `mlr_learners_clust.birch`, `mlr_learners_clust.clara`,
`mlr_learners_clust.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbs`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.flexmix`, `mlr_learners_clust.genie`, `mlr_learners_clust.gmeans`,
`mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kcca`, `mlr_learners_clust.kkme`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.kmeans_rcpp`, `mlr_learners_clust.kmodes`,
`mlr_learners_clust.kproto`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.movMF`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.protocl`
`mlr_learners_clust.skmeans`, `mlr_learners_clust.som`, `mlr_learners_clust.specc`, `mlr_learners_clust.tclust`
`mlr_learners_clust.xmeans`

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.stdbscan")
print(learner)
```

`mlr_learners_clust.tclust`

Robust Trimmed Clustering Learner

Description

Robust trimmed clustering. Each cluster is modeled by a multivariate Gaussian; the most outlying `alpha` fraction of observations is trimmed and labeled with cluster `0` in the returned partition. Calls `tclust::tclust()` from package **tclust**.

The `k` parameter is set to 2 by default since `tclust::tclust()` doesn't have a default value for the number of clusters. There is no predict method for `tclust::tclust()`, so the method returns cluster labels for the training data.

Setting `n.cores` to a value greater than one enables `parallel` at train time unless it is set explicitly.

Initial parameter values

- `store_x`:
  - Actual default: `TRUE`.
  - Adjusted default: `FALSE`.
  - Reason for change: Avoid storing the training data in the model to save memory.
- `n.cores`:
  - Actual default: `-1L`, using all available cores.
  - Adjusted default: `1L`.
  - Reason for change: Conflicting with parallelization via **future**.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.tclust")
lrn("clust.tclust")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3cluster**, **tclust**

Parameters

Id	Type	Default	Levels	Range
k	integer	-		$[1, \infty)$
alpha	numeric	0.05		$[0, 0.5]$
nstart	integer	500		$[1, \infty)$
niter1	integer	3		$[1, \infty)$
niter2	integer	20		$[1, \infty)$
nkeep	integer	5		$[1, \infty)$
equal.weights	logical	FALSE	TRUE, FALSE	-
restr	character	eigen	eigen, deter, GPCM	-
restr.fact	numeric	12		$[1, \infty)$
cshape	numeric	1e+10		$[1, \infty)$
pars	untyped	-		-
opt	character	HARD	HARD, MIXT	-
center	logical	FALSE	TRUE, FALSE	-
scale	logical	FALSE	TRUE, FALSE	-
store_x	logical	TRUE	TRUE, FALSE	-
parallel	logical	FALSE	TRUE, FALSE	-
n.cores	integer	-1		$[-2, \infty)$
zero_tol	numeric	1e-16		$[0, \infty)$
drop.empty.clust	logical	TRUE	TRUE, FALSE	-
trace	integer	0		$[0, 1]$

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustTclust`

Methods

Public methods:

- `LearnerClustTclust$new()`
- `LearnerClustTclust$clone()`

`LearnerClustTclust$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustTclust$new()
```

`LearnerClustTclust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustTclust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

García-Escudero, A L, Gordaliza, Alfonso, Matrán, Carlos, Mayo-Iscar, Agustín (2008). "A general trimming approach to robust cluster analysis." *The Annals of Statistics*, **36**(3), 1324–1345. doi:10.1214/07AOS515.

Fritz, Heinrich, García-Escudero, A L, Mayo-Iscar, Agustín (2012). "tclust: An R package for a trimming approach to cluster analysis." *Journal of Statistical Software*, **47**(12), 1–26. doi:10.18637/jss.v047.i12.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbs
mlr_learners_clust.xmeans

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.tclust")
print(learner)

# Define a Task
task = tsk("usarrests")

# Train the learner on the task
learner$train(task)
```

```
# Print the model
print(learner$native_model)

# Make predictions for the task
prediction = learner$predict(task)

# Score the predictions
prediction$score(task = task)
```

`mlr_learners_clust.xmeans`
X-Means Clustering Learner

Description

X-means clustering. Calls `RWeka::XMeans()` from package **RWeka**.

The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Installation

XMeans is not part of the Weka core and must be installed once through the Weka package manager:

```
RWeka::WPM("refresh-cache")
RWeka::WPM("install-package", "XMeans")
```

Initial parameter values

- `C`:
  - Actual default: `0.5` if no option is set, `0` otherwise.
  - Adjusted default: `0.5`.
  - Reason for change: Always pass `C` so the cutoff factor does not depend on other parameters.

Dictionary

This mlr3::Learner can be instantiated via the dictionary mlr3::mlr_learners or with the associated sugar function `mlr3::lrn()`:

```
mlr_learners$get("clust.xmeans")
lrn("clust.xmeans")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3cluster**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
B	numeric	1		$[0, \infty)$
C	numeric	-		$[0, \infty)$
D	untyped	"weka.core.EuclideanDistance"		-
H	integer	4		$[1, \infty)$
I	integer	1		$[1, \infty)$
J	integer	1000		$[1, \infty)$
K	untyped	""		-
L	integer	2		$[1, \infty)$
M	integer	1000		$[1, \infty)$
S	integer	10		$[0, \infty)$
U	integer	0		$[0, \infty)$
use_kdtree	logical	FALSE	TRUE, FALSE	-
N	untyped	-		-
O	untyped	-		-
Y	untyped	-		-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `LearnerClust` -> `LearnerClustXMeans`

Methods

Public methods:

- `LearnerClustXMeans$new()`
- `LearnerClustXMeans$clone()`

`LearnerClustXMeans$new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerClustXMeans$new()
```

`LearnerClustXMeans$clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerClustXMeans$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Witten, H I, Frank, Eibe (2002). "Data mining: practical machine learning tools and techniques with Java implementations." *Acm Sigmod Record*, **31**(1), 76–77.

Pelleg, Dan, Moore, W A, others (2000). "X-means: Extending k-means with efficient estimation of the number of clusters." In *Icml*, volume 1, 727–734.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html#sec-learners
- Package mlr3extralearners for more learners.
- Dictionary of Learners: mlr3::mlr_learners
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- Package **mlr3viz** for some generic visualizations.
- Extension packages for additional task types:
  - **mlr3proba** for probabilistic supervised regression and survival analysis.
  - **mlr3cluster** for unsupervised clustering.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Other Learner: mlr_learners_clust.MBatchKMeans, mlr_learners_clust.SimpleKMeans, mlr_learners_clust.agne
mlr_learners_clust.ap, mlr_learners_clust.bico, mlr_learners_clust.birch, mlr_learners_clust.clara,
mlr_learners_clust.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbs
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.flexmix, mlr_learners_clust.genie, mlr_learners_clust.gmeans,
mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kcca, mlr_learners_clust.kkme
mlr_learners_clust.kmeans, mlr_learners_clust.kmeans_rcpp, mlr_learners_clust.kmodes,
mlr_learners_clust.kproto, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.movMF, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.protocl
mlr_learners_clust.skmeans, mlr_learners_clust.som, mlr_learners_clust.specc, mlr_learners_clust.stdbsc
mlr_learners_clust.tclust

Examples

```
# Define the Learner and set parameter values
learner = lrn("clust.xmeans")
print(learner)
```

`mlr_measures_clust.avg_between` *Average Between-Cluster Distance*

Description

The mean of all pairwise distances between observations belonging to different clusters. Higher values indicate greater separation between clusters. This measure is scale-dependent and is most useful for comparing clusterings of the same dataset.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.avg_between")
msr("clust.avg_between")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bo`
`mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pears`
`mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`,
`mlr_measures_clust.wss`

`mlr_measures_clust.avg_within` *Average Within-Cluster Distance*

Description

The weighted mean of average pairwise distances within each cluster, where weights are the cluster sizes. Lower values indicate more compact clusters. This measure is scale-dependent and is most useful for comparing clusterings of the same dataset.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.avg_within")
msr("clust.avg_within")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: TRUE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_b`
`mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pears`
`mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`,
`mlr_measures_clust.wss`

`mlr_measures_clust.ch` *Calinski Harabasz Pseudo F-Statistic*

Description

The Calinski-Harabasz index (also known as the Variance Ratio Criterion) is the ratio of between-cluster variance to within-cluster variance, adjusted for the number of clusters and observations. It is defined as $CH = \frac{\mathrm{tr}(B)/(k-1)}{\mathrm{tr}(W)/(n-k)}$ where B is the between-cluster scatter matrix, W is the within-cluster scatter matrix, k is the number of clusters, and n is the number of observations. Higher values indicate better-defined clusters.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.ch")
msr("clust.ch")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**

References

Caliński, Tadeusz, Harabasz, Jerzy (1974). "A dendrite method for cluster analysis." *Communications in Statistics*, **3**(1), 1–27. doi:10.1080/03610927408827101.

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.davies_bouldin` *Davies-Bouldin Index*

Description

The Davies-Bouldin index measures the average similarity between each cluster and the cluster most similar to it, where similarity is the ratio of within-cluster scatter to between-cluster separation. It is defined as $DB = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i} \frac{s_i + s_j}{d_{ij}}$ where s_i is the average distance of observations in cluster i to its centroid and d_{ij} is the Euclidean distance between centroids i and j. Lower values indicate better clustering.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.davies_bouldin")
msr("clust.davies_bouldin")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: TRUE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**

References

Davies, L D, Bouldin, W D (1979). "A cluster separation measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1**(2), 224–227. doi:10.1109/TPAMI.1979.4766909.

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.dunn` *Dunn Index*

Description

The Dunn index is the ratio of the smallest inter-cluster distance to the largest intra-cluster diameter, defined as $D = \min_{i \neq j} \delta(C_i, C_j) / \max_k \Delta(C_k)$ where $\delta(C_i, C_j)$ is the minimum distance between clusters i and j, and $\Delta(C_k)$ is the maximum distance between any two observations in cluster k. Higher values indicate compact, well-separated clusters.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.dunn")
msr("clust.dunn")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

References

Dunn, C J (1974). "Well-separated clusters and optimal fuzzy partitions." *Journal of Cybernetics*, **4**(1), 95–104. doi:10.1080/01969727408546059.

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.dunn2` *Dunn2 Index*

Description

An alternative formulation of the Dunn index that uses average distances instead of extremes. It is defined as the ratio of the minimum average between-cluster distance to the maximum average within-cluster distance: $D_2 = \min_{i \neq j} \bar{d}(C_i, C_j) / \max_k \bar{d}(C_k)$. This variant is more robust to outliers than the standard Dunn index. Higher values indicate better separation.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.dunn2")
msr("clust.dunn2")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

References

Dunn, C J (1974). "Well-separated clusters and optimal fuzzy partitions." *Journal of Cybernetics*, **4**(1), 95–104. doi:10.1080/01969727408546059.

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.entropy` *Entropy*

Description

The Shannon entropy of the cluster size distribution, defined as $H = -\sum_{k=1}^{K} p_k \log(p_k)$ where $p_k = n_k/n$ is the proportion of observations in cluster k. Lower values indicate more uneven cluster sizes (with 0 for a single cluster), while higher values indicate more uniform sizes. This measure does not evaluate cluster quality directly but characterizes the balance of the partition.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.entropy")
msr("clust.entropy")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: NA
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.pearsongamma` *Pearson Gamma*

Description

The Pearson correlation between pairwise distances and a binary indicator of whether two observations belong to different clusters. All within-cluster distances are paired with indicator 0, and all between-cluster distances with indicator 1. Values close to 1 indicate that between-cluster distances tend to be larger than within-cluster distances, suggesting well-separated clusters.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.pearsongamma")
msr("clust.pearsongamma")
```

Meta Information

- Task type: "clust"
- Range: $[-1, 1]$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.silhouette`

Rousseeuw's Silhouette Quality Index

Description

Calls `cluster::silhouette()` from package **cluster**.

The Silhouette Width measures how well each observation fits within its assigned cluster compared to neighboring clusters. For each observation, the silhouette value is defined as $s(i) = (b(i) - a(i))/\max(a(i), b(i))$ where $a(i)$ is the average distance to all other observations in the same cluster and $b(i)$ is the minimum average distance to observations in any other cluster. The score returned is the mean silhouette width across all observations. Values close to 1 indicate well-clustered observations, values near 0 indicate observations on cluster boundaries, and negative values indicate possible misclassification.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.silhouette")
msr("clust.silhouette")
```

Meta Information

- Task type: "clust"
- Range: $[-1, 1]$
- Minimize: FALSE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

References

Rousseeuw, J P (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics*, **20**, 53–65. doi:10.1016/0377-0427(87)901257.

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.sse_ratio`

Within/Total Sum of Squares Ratio

Description

The ratio of the within-cluster sum of squares to the total sum of squares, $WSS/TSS = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2 / \sum_{i=1}^{n} \|x_i - \mu\|^2$ where μ_k is the centroid of cluster k and μ is the centroid of all observations. The ratio lies in $[0, 1]$, with lower values indicating more compact clusters. Unlike the raw within sum of squares, it is comparable across datasets and feature scalings. It decreases monotonically with the number of clusters, which makes it the standard quantity for elbow plots.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.sse_ratio")
msr("clust.sse_ratio")
```

Meta Information

- Task type: "clust"
- Range: $[0, 1]$
- Minimize: TRUE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.wb_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.wb_ratio`

Within/Between Ratio

Description

The ratio of the average within-cluster distance to the average between-cluster distance. The average within-cluster distance is the weighted mean of all pairwise distances within each cluster, and the average between-cluster distance is the mean of all pairwise distances between observations in different clusters. Lower values indicate compact clusters that are well separated from each other.

Details

If the task contains character, factor, or ordered features, Gower distances (`cluster::daisy()`) are used instead of Euclidean distances.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.wb_ratio")
msr("clust.wb_ratio")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: TRUE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**, **cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wss`

`mlr_measures_clust.wss`

Within Sum of Squares

Description

The total within-cluster sum of squares measures the compactness of the clustering by summing the squared Euclidean distances of each observation to its cluster centroid across all clusters: $WSS = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$. Lower values indicate tighter clusters.

Dictionary

This mlr3::Measure can be instantiated via the dictionary mlr3::mlr_measures or with the associated sugar function `mlr3::msr()`:

```
mlr_measures$get("clust.wss")
msr("clust.wss")
```

Meta Information

- Task type: "clust"
- Range: $[0, \infty)$
- Minimize: TRUE
- Average: macro
- Required Prediction: "partition"
- Required Packages: **mlr3**, **mlr3cluster**

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.avg_between`, `mlr_measures_clust.avg_within`, `mlr_measures_clust.ch`, `mlr_measures_clust.davies_bouldin`, `mlr_measures_clust.dunn`, `mlr_measures_clust.dunn2`, `mlr_measures_clust.entropy`, `mlr_measures_clust.pearsongamma`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.sse_ratio`, `mlr_measures_clust.wb_ratio`

`mlr_tasks_ruspini` *Ruspini Cluster Task*

Description

A cluster task for the cluster::ruspini data set.

Format

R6::R6Class inheriting from TaskClust.

Dictionary

This mlr3::Task can be instantiated via the dictionary mlr3::mlr_tasks or with the associated sugar function `mlr3::tsk()`:

```
mlr_tasks$get("ruspini")
tsk("ruspini")
```

Meta Information

- Task type: "clust"
- Dimensions: 75x2
- Properties: -
- Has Missings: `FALSE`
- Target: -
- Features: "x", "y"

References

Ruspini EH (1970). "Numerical methods for fuzzy clustering." *Information Sciences*, **2**(3), 319-350. doi:10.1016/S00200255(70)800561.

See Also

- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html`
- Package **mlr3data** for more toy tasks.
- Package **mlr3oml** for downloading tasks from `https://www.openml.org`.
- Package **mlr3viz** for some generic visualizations.
- Dictionary of Tasks: mlr3::mlr_tasks
- `as.data.table(mlr_tasks)` for a table of available Tasks in the running session (depending on the loaded packages).
- **mlr3fselect** and **mlr3filters** for feature selection and feature filtering.

- Extension packages for additional task types:
  - Unsupervised clustering: **mlr3cluster**
  - Probabilistic supervised regression and survival analysis: https://mlr3proba.mlr-org.com/.

Other Task: TaskClust, mlr_tasks_usarrests

`mlr_tasks_usarrests` *US Arrests Cluster Task*

Description

A cluster task for the datasets::USArrests data set. Rownames are stored as variable `"state"` with column role `"name"`.

Format

R6::R6Class inheriting from TaskClust.

Dictionary

This mlr3::Task can be instantiated via the dictionary mlr3::mlr_tasks or with the associated sugar function `mlr3::tsk()`:

```
mlr_tasks$get("usarrests")
tsk("usarrests")
```

Meta Information

- Task type: "clust"
- Dimensions: 50x4
- Properties: -
- Has Missings: `FALSE`
- Target: -
- Features: "Assault", "Murder", "Rape", "UrbanPop"

References

Berry, Brian J (1979). "Interactive Data Analysis: A Practical Primer." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **28**, 181.

See Also

- Chapter in the mlr3book: https://mlr3book.mlr-org.com/chapters/chapter2/data_and_basic_modeling.html
- Package **mlr3data** for more toy tasks.
- Package **mlr3oml** for downloading tasks from https://www.openml.org.
- Package **mlr3viz** for some generic visualizations.
- Dictionary of Tasks: mlr3::mlr_tasks
- `as.data.table(mlr_tasks)` for a table of available Tasks in the running session (depending on the loaded packages).
- **mlr3fselect** and **mlr3filters** for feature selection and feature filtering.
- Extension packages for additional task types:
  - Unsupervised clustering: **mlr3cluster**
  - Probabilistic supervised regression and survival analysis: https://mlr3proba.mlr-org.com/.

Other Task: TaskClust, mlr_tasks_ruspini

`mlr_task_generators_blobs`

Gaussian Blobs Cluster Task Generator

Description

A TaskGenerator for isotropic Gaussian blobs, in the spirit of `sklearn.datasets.make_blobs()`. `k` cluster centers are drawn uniformly from the hypercube `[-center_box, center_box]^d`, and the `n` observations are assigned to the centers in a balanced fashion and perturbed with Gaussian noise of standard deviation `sd` in each of the `d` dimensions. The generated TaskClust only contains the numeric features `x1`, ..., `xd`; the cluster membership is not stored in the task. The parameters are initialized to `k = 3`, `d = 2`, `sd = 1`, and `center_box = 10`.

Dictionary

This TaskGenerator can be instantiated via the dictionary mlr_task_generators or with the associated sugar function tgen():

```
mlr_task_generators$get("blobs")
tgen("blobs")
```

Parameters

Id	Type	Default	Range
k	integer	-	$[1, \infty)$
d	integer	-	$[1, \infty)$
sd	numeric	-	$[0, \infty)$
center_box	numeric	-	$[0, \infty)$

Super class

`mlr3::TaskGenerator` -> `TaskGeneratorBlobs`

Methods

Public methods:

- `TaskGeneratorBlobs$new()`
- `TaskGeneratorBlobs$plot()`
- `TaskGeneratorBlobs$clone()`

`TaskGeneratorBlobs$new()`: Creates a new instance of this R6 class.

Usage:

```
TaskGeneratorBlobs$new()
```

`TaskGeneratorBlobs$plot()`: Creates a simple plot of the first two features of generated data, colored by cluster membership.

Usage:

```
TaskGeneratorBlobs$plot(n = 200L, pch = 19L, ...)
```

Arguments:

`n (integer(1))`
: Number of samples to draw for the plot. Default is `200`.

`pch (integer(1))`
: Point char. Passed to `graphics::plot()`.

`...` (any)
: Additional arguments passed to `graphics::plot()`.

`TaskGeneratorBlobs$clone()`: The objects of this class are cloneable with this method.

Usage:

```
TaskGeneratorBlobs$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

- Dictionary of TaskGenerators: mlr3::mlr_task_generators
- `as.data.table(mlr_task_generators)` for a table of available TaskGenerators in the running session (depending on the loaded packages).

Examples

```
generator = tgen("blobs")
plot(generator, n = 200)

task = generator$generate(200)
str(task$data())

# 4 well separated clusters in 3 dimensions
generator = tgen("blobs", k = 4, d = 3, sd = 0.5)
task = generator$generate(500)
task
```

`PredictionClust` *Prediction Object for Cluster Analysis*

Description

This object wraps the predictions returned by a learner of class LearnerClust, i.e. the predicted partition and cluster probability.

Super class

`mlr3::Prediction` -> `PredictionClust`

Active bindings

`partition` (`integer()`)
Access the stored partition.

`prob` (`matrix()` | `NULL`)
Access to the stored probabilities.

Methods

Public methods:

- `PredictionClust$new()`
- `PredictionClust$clone()`

`PredictionClust$new()`: Creates a new instance of this R6 class.

Usage:

```
PredictionClust$new(
  task = NULL,
  row_ids = task$row_ids,
  partition = NULL,
  prob = NULL,
  weights = NULL,
  check = TRUE,
  extra = NULL,
  raw = NULL
)
```

Arguments:

`task` (TaskClust | `NULL`)
: Task, used to extract defaults for `row_ids`.

`row_ids` (`integer()`)
: Row ids of the predicted observations, i.e. the row ids of the test set.

`partition` (`integer()` | `NULL`)
: Vector of cluster partitions.

`prob` (`matrix()` | `NULL`)
: Numeric matrix of cluster membership probabilities with one column for each cluster and one row for each observation. Columns must be named with cluster numbers, row names are automatically removed. If `prob` is provided, but `partition` is not, the cluster memberships are calculated from the probabilities: each observation is assigned the cluster label (column name) of its most probable cluster, with ties broken by the first maximum.

`weights` (`numeric()`)
: Vector of measure weights for each observation. Should be constructed from the TaskClust's `weights_measure` column.

`check` (`logical(1)`)
: If `TRUE`, performs some argument checks and predict type conversions.

`extra` (`list()`)
: List of extra data to be stored in the prediction object.

`raw` (any)
: Raw prediction object from the upstream model. Stored as-is without validation.

`PredictionClust$clone()`: The objects of this class are cloneable with this method.

Usage:

```
PredictionClust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(mlr3)
library(mlr3cluster)
task = tsk("usarrests")
learner = lrn("clust.featureless", num_clusters = 2L)
p = learner$train(task)$predict(task)
p$predict_types
head(as.data.table(p))
```

TaskClust *Cluster Task*

Description

This task specializes mlr3::Task for cluster problems. As an unsupervised task, this task has no target column. The task_type is set to "clust".

Predefined tasks are stored in the dictionary mlr3::mlr_tasks.

Super classes

mlr3::Task -> mlr3::TaskUnsupervised -> TaskClust

Methods

Public methods:

- TaskClust$new()
- TaskClust$clone()

TaskClust$new(): Creates a new instance of this R6 class.

Usage:

```
TaskClust$new(id, backend, label = NA_character_, extra_args = list())
```

Arguments:

id (character(1))
: Identifier for the new instance.

backend (mlr3::DataBackend)
: Either a mlr3::DataBackend, or any object which is convertible to a mlr3::DataBackend with as_data_backend(). E.g., a data.frame() will be converted to a mlr3::DataBackendDataTable.

label (character(1))
: Label for the new instance.

extra_args (named list())
: Named list of constructor arguments, required for converting task types via mlr3::convert_task().

TaskClust$clone(): The objects of this class are cloneable with this method.

Usage:

```
TaskClust$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other Task: mlr_tasks_ruspini, mlr_tasks_usarrests

Examples

```
library(mlr3)
library(mlr3cluster)
task = TaskClust$new("usarrests", backend = USArrests)
task$task_type

# possible properties:
mlr_reflections$task_properties$clust
```

Index

∗ **Learner**
mlr_learners_clust.agnes, 12
mlr_learners_clust.ap, 15
mlr_learners_clust.bico, 18
mlr_learners_clust.birch, 20
mlr_learners_clust.clara, 22
mlr_learners_clust.cmeans, 25
mlr_learners_clust.cobweb, 28
mlr_learners_clust.dbscan, 31
mlr_learners_clust.dbscan_fpc, 33
mlr_learners_clust.diana, 35
mlr_learners_clust.em, 38
mlr_learners_clust.fanny, 41
mlr_learners_clust.featureless, 44
mlr_learners_clust.ff, 46
mlr_learners_clust.flexmix, 49
mlr_learners_clust.genie, 52
mlr_learners_clust.gmeans, 54
mlr_learners_clust.hclust, 57
mlr_learners_clust.hdbscan, 60
mlr_learners_clust.kcca, 62
mlr_learners_clust.kkmeans, 65
mlr_learners_clust.kmeans, 68
mlr_learners_clust.kmeans_rcpp, 70
mlr_learners_clust.kmodes, 73
mlr_learners_clust.kproto, 76
mlr_learners_clust.MBatchKMeans, 79
mlr_learners_clust.mclust, 81
mlr_learners_clust.meanshift, 84
mlr_learners_clust.movMF, 87
mlr_learners_clust.optics, 90
mlr_learners_clust.pam, 92
mlr_learners_clust.protoclust, 95
mlr_learners_clust.SimpleKMeans, 98
mlr_learners_clust.skmeans, 101
mlr_learners_clust.som, 104
mlr_learners_clust.specc, 107
mlr_learners_clust.stdbscan, 109
mlr_learners_clust.tclust, 112
mlr_learners_clust.xmeans, 115
∗ **Prediction**
PredictionClust, 134
∗ **TaskGenerator**
mlr_task_generators_blobs, 132
∗ **Task**
mlr_tasks_ruspini, 130
mlr_tasks_usarrests, 131
TaskClust, 136
∗ **cluster measures**
mlr_measures_clust.avg_between, 118
mlr_measures_clust.avg_within, 119
mlr_measures_clust.ch, 120
mlr_measures_clust.davies_bouldin, 121
mlr_measures_clust.dunn, 122
mlr_measures_clust.dunn2, 123
mlr_measures_clust.entropy, 124
mlr_measures_clust.pearsongamma, 125
mlr_measures_clust.silhouette, 126
mlr_measures_clust.sse_ratio, 127
mlr_measures_clust.wb_ratio, 128
mlr_measures_clust.wss, 129

apcluster::apcluster(), *15, 16*
as_prediction_clust, 4
as_task_clust, 5
as_tasks_clust *(*as_task_clust*)*, 5

benchmark(), *11*

clue::cl_predict(), *22*, *25*, *68*, *92*
clust.dunn, *12*
cluster::agnes(), *12*
cluster::clara(), *22*

cluster::daisy(), *118, 119*, *122, 123*, *125, 126*, *128*
cluster::diana(), *35*
cluster::fanny(), *41*
cluster::pam(), *92*
cluster::ruspini, *130*
cluster::silhouette(), *126*
ClusterR::KMeans_rcpp(), *70*
ClusterR::MiniBatchKmeans(), *79*
ClusterR::predict_KMeans(), *79*
clustMixType::kproto(), *76*

data.frame(), *5*
datasets::USArrests, *131*
dbscan::dbscan(), *31*
dbscan::extractDBSCAN(), *90*
dbscan::hdbscan(), *60*
dbscan::optics(), *90*
Dictionary, *14*, *17*, *19*, *22*, *24*, *27*, *30*, *32*, *34*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *69*, *72*, *75*, *78*, *80*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *108*, *111*, *114*, *117–125*, *127–130*, *132*, *134*
dictionary, *13*, *15*, *18*, *20*, *23*, *25*, *28*, *31*, *33*, *36*, *38*, *42*, *44*, *46*, *49*, *52*, *54*, *57*, *60*, *62*, *65*, *68*, *71*, *73*, *76*, *79*, *82*, *85*, *87*, *90*, *92*, *95*, *98*, *101*, *104*, *107*, *110*, *112*, *115*, *118–132*, *136*

e1071::cmeans(), *25*

flexclust::kcca(), *62*
flexmix::flexmix(), *49*
flexmix::stepFlexmix(), *49*
formula, *5*
fpc::dbscan(), *33, 34*

genieclust::gclust(), *52*
gmeans::gmeans(), *54*
graphics::plot(), *133*

kernlab::kkmeans(), *65, 66*
kernlab::specc(), *107*
klaR::kmodes(), *73*
kohonen::map(), *104*
kohonen::som(), *104*
kohonen::somgrid(), *104*

LearnerClust, 7, *13*, *16*, *18*, *21*, *23*, *26*, *29*, *31*, *34*, *36*, *39*, *42*, *45*, *47*, *50*, *53*, *55*, *58*, *60*, *63*, *66*, *68*, *71*, *74*, *77*, *80*, *82*, *85*, *88*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *110*, *113*, *116*, *134*
LearnerClustAgnes *(*mlr_learners_clust.agnes*)*, 12
LearnerClustAP *(*mlr_learners_clust.ap*)*, 15
LearnerClustBICO *(*mlr_learners_clust.bico*)*, 18
LearnerClustBIRCH *(*mlr_learners_clust.birch*)*, 20
LearnerClustCLARA *(*mlr_learners_clust.clara*)*, 22
LearnerClustCMeans *(*mlr_learners_clust.cmeans*)*, 25
LearnerClustCobweb *(*mlr_learners_clust.cobweb*)*, 28
LearnerClustDBSCAN *(*mlr_learners_clust.dbscan*)*, 31
LearnerClustDBSCANfpc *(*mlr_learners_clust.dbscan_fpc*)*, 33
LearnerClustDiana *(*mlr_learners_clust.diana*)*, 35
LearnerClustEM *(*mlr_learners_clust.em*)*, 38
LearnerClustFanny *(*mlr_learners_clust.fanny*)*, 41
LearnerClustFarthestFirst *(*mlr_learners_clust.ff*)*, 46
LearnerClustFeatureless *(*mlr_learners_clust.featureless*)*, 44
LearnerClustFlexmix *(*mlr_learners_clust.flexmix*)*, 49
LearnerClustGenie *(*mlr_learners_clust.genie*)*, 52
LearnerClustGMeans *(*mlr_learners_clust.gmeans*)*, 54
LearnerClustHclust *(*mlr_learners_clust.hclust*)*, 57
LearnerClustHDBSCAN *(*mlr_learners_clust.hdbscan*)*, 60
LearnerClustKCCA *(*mlr_learners_clust.kcca*)*, 62
LearnerClustKKMeans

(mlr_learners_clust.kkmeans), 65
LearnerClustKMeans (mlr_learners_clust.kmeans), 68
LearnerClustKMeansRcpp (mlr_learners_clust.kmeans_rcpp), 70
LearnerClustKModes (mlr_learners_clust.kmodes), 73
LearnerClustKProto (mlr_learners_clust.kproto), 76
LearnerClustMclust (mlr_learners_clust.mclust), 81
LearnerClustMeanShift (mlr_learners_clust.meanshift), 84
LearnerClustMiniBatchKMeans (mlr_learners_clust.MBatchKMeans), 79
LearnerClustMovMF (mlr_learners_clust.movMF), 87
LearnerClustOPTICS (mlr_learners_clust.optics), 90
LearnerClustPAM (mlr_learners_clust.pam), 92
LearnerClustProtoclust (mlr_learners_clust.protoclust), 95
LearnerClustSimpleKMeans (mlr_learners_clust.SimpleKMeans), 98
LearnerClustSKMeans (mlr_learners_clust.skmeans), 101
LearnerClustSOM (mlr_learners_clust.som), 104
LearnerClustSpectral (mlr_learners_clust.specc), 107
LearnerClustSTDBSCAN (mlr_learners_clust.stdbscan), 109
LearnerClustTclust (mlr_learners_clust.tclust), 112
LearnerClustXMeans (mlr_learners_clust.xmeans), 115
Learners, *14*, *17*, *19*, *22*, *24*, *27*, *30*, *32*, *34*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *69*, *72*, *75*, *78*, *80*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *108*, *111*, *114*, *117*
LPCM::ms(), *84*, *85*
LPCM::ms.rep(), *84*

matrix(), *5*
mclust::Mclust(), *81*
mclust::predict.Mclust(), *81*
mean(), *11*
MeasureClust, 10
Measures, *118–125*, *127–129*
mlr3::convert_task(), *5*, *6*, *136*
mlr3::DataBackend, *5*, *136*
mlr3::DataBackendDataTable, *136*
mlr3::Learner, *7–9*, *11*, *13*, *15*, *16*, *18*, *20*, *21*, *23*, *25*, *26*, *28*, *29*, *31*, *33*, *34*, *36*, *38*, *39*, *42*, *44–47*, *49*, *50*, *52–55*, *57*, *58*, *60*, *62*, *63*, *65*, *66*, *68*, *71*, *73*, *74*, *76*, *77*, *79*, *80*, *82*, *85*, *87*, *88*, *90*, *92*, *93*, *95*, *96*, *98*, *99*, *101*, *102*, *104*, *105*, *107*, *108*, *110*, *112*, *113*, *115*, *116*
mlr3::lrn(), *13*, *15*, *18*, *20*, *23*, *25*, *28*, *31*, *33*, *36*, *38*, *42*, *44*, *46*, *49*, *52*, *54*, *57*, *60*, *62*, *65*, *68*, *71*, *73*, *76*, *79*, *82*, *85*, *87*, *90*, *92*, *95*, *98*, *101*, *104*, *107*, *110*, *112*, *115*
mlr3::marshal_model(), *29*, *40*, *47*, *99*
mlr3::Measure, *10*, *118–129*
mlr3::mlr_learners, *7*, *13–15*, *17–20*, *22–25*, *27*, *28*, *30–34*, *36–38*, *40*, *42–46*, *48*, *49*, *51–54*, *56*, *57*, *59–62*, *64*, *65*, *67–69*, *71–73*, *75*, *76*, *78–80*, *82*, *83*, *85–87*, *89–92*, *94*, *95*, *97*, *98*, *100*, *101*, *103*, *104*, *106–108*, *110–112*, *114*, *115*, *117*
mlr3::mlr_measures, *10*, *118–129*
mlr3::mlr_task_generators, *134*
mlr3::mlr_tasks, *130–132*, *136*
mlr3::msr(), *118–129*
mlr3::Prediction, *7*, *11*, *134*
mlr3::Resampling, *11*
mlr3::Task, *11*, *12*, *130*, *131*, *136*
mlr3::TaskClassif, *5*
mlr3::TaskGenerator, *133*
mlr3::TaskRegr, *5*
mlr3::TaskSupervised, *5*

mlr3::TaskUnsupervised, *136*
mlr3::tsk(), *130, 131*
mlr3::unmarshal_model(), *29*, *40*, *47*, *99*
mlr3cluster *(*mlr3cluster-package*)*, 4
mlr3cluster-package, 4
mlr3misc::Dictionary, *7*, *10*
mlr_learners_clust.agnes, 12, *17*, *19*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.ap, *14*, 15, *19*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.bico, *14*, *17*, 18, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.birch, *14*, *17*, *19*, 20, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.clara, *14*, *17*, *19*, *22*, 22, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.cmeans, *14*, *17*, *19*, *22*, *24*, 25, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.cobweb, *14*, *17*, *19*, *22*, *24*, *27*, 28, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.dbscan, *14*, *17*, *19*, *22*, *24*, *27*, *30*, 31, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.dbscan_fpc, *14*, *17*, *19*, *22*, *24*, *27*, *30*, *32*, 33, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.diana, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, 35, *40*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.em, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, 38, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.fanny, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, 41, *45*, *48*, *51*, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.featureless, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, 44, *48*, *51*, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.ff, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, 46, *51*, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.flexmix, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, 49, *54*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.genie, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, 52, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.gmeans, *14*, *17*, *20*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *54*, 54, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.hclust, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *54*, *56*, 57, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*

mlr_learners_clust.hdbscan, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, 60, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kcca, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, 62, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kkmeans, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, 65, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kmeans, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, 68, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kmeans_rcpp, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, 70, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kmodes, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, 73, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.kproto, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, 76, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.MBatchKMeans, *14*, *17*, *19*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, 79, *83*, *86*, *89*, *91*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.mclust, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, 81, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.meanshift, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, 84, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.movMF, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, 87, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.optics, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, 90, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.pam, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, 92, *97*, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.protoclust, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, 95, *100*, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.SimpleKMeans, *14*, *17*, *19*, *22*, *24*, *27*, *30*, *32*, *35*, *37*, *40*, *43*, *45*, *48*, *51*, *53*, *56*, *59*, *61*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *83*, *86*, *89*, *91*, *94*, *97*, 98, *103*, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.skmeans, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, 101, *106*, *109*, *111*, *114*, *117*
mlr_learners_clust.som, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, 104, *109*, *111*, *114*, *117*
mlr_learners_clust.specc, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, 107, *111*, *114*, *117*
mlr_learners_clust.stdbscan, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, 109, *114*, *117*
mlr_learners_clust.tclust, *14*, *17*, *20*, *22*,

25, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, 112, *117*
mlr_learners_clust.xmeans, *14*, *17*, *20*, *22*, *25*, *27*, *30*, *33*, *35*, *37*, *41*, *43*, *45*, *48*, *51*, *54*, *56*, *59*, *62*, *64*, *67*, *70*, *72*, *75*, *78*, *81*, *84*, *86*, *89*, *92*, *94*, *97*, *100*, *103*, *106*, *109*, *111*, *114*, 115
mlr_measures_clust.avg_between, 118, *119–125*, *127–129*
mlr_measures_clust.avg_within, *118*, 119, *120–125*, *127–129*
mlr_measures_clust.ch, *118*, *119*, 120, *121–125*, *127–129*
mlr_measures_clust.davies_bouldin, *118–120*, 121, *122–125*, *127–129*
mlr_measures_clust.dunn, *118–121*, 122, *123–125*, *127–129*
mlr_measures_clust.dunn2, *118–122*, 123, *124*, *125*, *127–129*
mlr_measures_clust.entropy, *118–123*, 124, *125*, *127–129*
mlr_measures_clust.pearsongamma, *118–124*, 125, *127–129*
mlr_measures_clust.silhouette, *118–125*, 126, *128*, *129*
mlr_measures_clust.sse_ratio, *118–125*, *127*, 127, *129*
mlr_measures_clust.wb_ratio, *118–125*, *127*, *128*, 128, *129*
mlr_measures_clust.wss, *118–125*, *127–129*, 129
mlr_reflections$learner_predict_types, *8*, *11*
mlr_reflections$learner_properties, *8*
mlr_reflections$measure_properties, *11*
mlr_reflections$task_feature_types, *8*
mlr_task_generators, *132*
mlr_task_generators_blobs, 132
mlr_tasks_ruspini, 130, *132*, *136*
mlr_tasks_usarrests, *131*, 131, *136*
movMF::movMF(), *87*

paradox::ParamSet, *8*, *10*
PredictionClust, *4*, *5*, *7*, 134
protoclust::protoclust(), *95*, *96*
protoclust::protocut(), *95*

R6, *8*, *10*, *14*, *16*, *19*, *21*, *24*, *26*, *29*, *32*, *34*, *37*, *39*, *42*, *45*, *47*, *50*, *53*, *55*, *58*, *61*, *63*, *66*, *69*, *71*, *74*, *77*, *80*, *83*, *85*, *88*, *91*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *116*, *133*, *134*, *136*
R6::R6Class, *130*, *131*
requireNamespace(), *9*, *12*
resample(), *11*
ResampleResult, *11*
RWeka::Cobweb(), *28*
RWeka::FarthestFirst(), *46*
RWeka::predict.Weka_clusterer(), *28*, *38*, *46*, *98*, *115*
RWeka::SimpleKMeans(), *98*
RWeka::XMeans(), *115*

skmeans::skmeans(), *101*
stats::cutree(), *12*, *35*, *52*, *57*
stats::dist(), *57*, *95*
stats::hclust(), *57*
stats::kmeans(), *68*
stdbscan::st_dbscan(), *110*
stream::DSC_BICO(), *18*
stream::DSC_BIRCH(), *20*
stream::DSC_Kmeans(), *18*
stream::DSC_TwoStage(), *18*

TaskClust, *5*, *7*, *130–132*, *135*, 136
TaskGenerator, *132*
TaskGeneratorBlobs *(*mlr_task_generators_blobs*)*, 132
TaskGenerators, *134*
Tasks, *130*, *132*
tclust::tclust(), *112*
tgen(), *132*