Package 'mlr3cluster'

August 14, 2022

Title Cluster Extension for 'mlr3'

Version 0.1.4

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 R (>= 3.1.0), mlr3 (>= 0.14.0)

Imports backports (>= 1.1.10), checkmate, clue, clusterCrit, cluster, data.table, mlr3misc (>= 0.9.4), paradox, R6, stats

Suggests dbscan, e1071, ClusterR, kernlab, apcluster, LPCM, mlbench, RWeka, testthat

Encoding UTF-8

RoxygenNote 7.2.1

Collate 'LearnerClust.R' 'LearnerClustAffinityPropagation.R' 'LearnerClustAgnes.R' 'LearnerClustCMeans.R' 'LearnerClustCobweb.R' 'LearnerClustDBSCAN.R' 'LearnerClustDiana.R' 'LearnerClustEM.R' 'LearnerClustFanny.R' 'LearnerClustFarthestFirst.R' 'LearnerClustFeatureless.R' 'LearnerClustHclust.R' 'LearnerClustKKMeans.R' 'LearnerClustKMeans.R' 'LearnerClustMeanShift.R' 'LearnerClustMiniBatchKMeans.R' 'LearnerClustPAM.R' 'LearnerClustSimpleKMeans.R' 'LearnerClustXMeans.R' 'MeasureClust.R' 'measures.R' 'MeasureClustInternal.R' 'PredictionClust.R' 'PredictionDataClust.R' 'TaskClust.R' 'TaskClust_usarrest.R' 'as_prediction_clust.R' 'as_task_clust.R' 'helper.R' 'zzz.R'

NeedsCompilation no

Author Damir Pulatov [cre, aut],
Michel Lang [aut] (<`https://orcid.org/0000-0001-9754-0393`>)

Maintainer Damir Pulatov <`damirpolat@protonmail.com`>

Repository CRAN

Date/Publication 2022-08-14 19:40:05 UTC

R topics documented:

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

Description

Extends the 'mlr3' package with cluster analysis.

Author(s)

Maintainer: Damir Pulatov <damirpolat@protonmail.com>

Authors:

- Michel Lang <michellang@gmail.com> (ORCID)

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.

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

```
if (requireNamespace("e1071")) {
  # create a prediction object
  task = tsk("usarrests")
  learner = lrn("clust.kmeans")
  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. This is a S3 generic, specialized for at least the following objects:

1. TaskClust: ensure the identity.
2. `data.frame()` and DataBackend: provides an alternative to calling constructor of TaskClust.

Usage

```
as_task_clust(x, ...)

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

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

## S3 method for class 'DataBackend'
as_task_clust(x, id = deparse(substitute(x)), ...)
```

```
## S3 method for class 'formula'
as_task_clust(x, data, id = deparse(substitute(data)), ...)
```

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`.
`id`	(`character(1)`) Id for the new task. Defaults to the (deparsed and substituted) name of the data argument.
`data`	(`data.frame()`) Data frame containing all columns specified in formula `x`.

Value

TaskClust.

Examples

```
as_task_clust(datasets::USArrests)
```

`LearnerClust` *Cluster Learner*

Description

This Learner specializes mlr3::Learner for cluster problems:

- `task_type` is set to `"clust"`.
- Creates Predictions of class PredictionClust.
- Possible values for `predict_types` are:
  - `"partition"`: Integer indicating the cluster membership.
  - `"prob"`: Probability for belonging to each cluster.

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.

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

Methods

Public methods:

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

Method `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_
)
```

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 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.
  - `"importance"`: The learner supports extraction of importance scores, i.e. comes with an `$importance()` extractor function (see section on optional extractors in Learner).
  - `"selected_features"`: The learner supports extraction of the set of selected features, i.e. comes with a `$selected_features()` extractor function (see section on optional extractors in 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 Learner).

`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.

Method `reset()`**:** Reset `assignments` field before calling parent's `reset()`.

Usage:

```
LearnerClust$reset()
```

Method `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:
lrn = mlr_learners$get("clust.kmeans")
print(lrn)
```

`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()`

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

Usage:

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

Arguments:

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

`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.

`aggregator (function(x))`
: Function to aggregate individual performance scores `x` where `x` is a numeric vector. If `NULL`, defaults to `mean()`.

`properties (character())`
: Properties of the measure. Must be a subset of mlr_reflections$measure_properties. Supported by `mlr3`:
  - `"requires_task"` (requires the complete Task),
  - `"requires_learner"` (requires the trained Learner),
  - `"requires_train_set"` (requires the training indices from the Resampling), and
  - `"na_score"` (the measure is expected to occasionally return `NA` or `NaN`).

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

`task_properties (character())`
: Required task properties, see 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()`.

See Also

Example cluster measures: `clust.dunn`

`mlr_learners_clust.agnes`
Agglomerative Hierarchical Clustering Learner

Description

A LearnerClust for agglomerative hierarchical clustering implemented in `cluster::agnes()`. The predict method uses `stats::cutree()` which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter `k`). The default number for `k` is 2.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	-
trace.lev	integer	0		$[0, \infty)$
k	integer	2		$[1, \infty)$
par.method	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustAgnes`

Methods

Public methods:

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

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

Usage:

```
LearnerClustAgnes$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("cluster")) {
  learner = mlr3::lrn("clust.agnes")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.ap` *Affinity Propagation Clustering Learner*

Description

A LearnerClust for Affinity Propagation clustering implemented in `apcluster::apcluster()`. `apcluster::apcluster()` doesn't have set a default for similarity function. Therefore, the `s` parameter here is set to `apcluster::negDistMat(r = 2L)` by default since this is what is used in the original paper on Affity Propagation clustering. 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.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	apcluster::negDistMat, 2		-
p	untyped	NA		-
q	numeric	-		$[0, 1]$
maxits	integer	1000		$[1, \infty)$
convits	integer	100		$[1, \infty)$
lam	numeric	0.9		$[0.5, 1]$
includeSim	logical	FALSE	TRUE, FALSE	-
details	logical	FALSE	TRUE, FALSE	-
nonoise	logical	FALSE	TRUE, FALSE	-
seed	integer	-		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustAP`

Methods

Public methods:

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

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

Usage:

```
LearnerClustAP$new()
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
if (requireNamespace("apcluster")) {
  learner = mlr3::lrn("clust.ap")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.cmeans` *Fuzzy C-Means Clustering Learner*

Description

A LearnerClust for fuzzy clustering implemented in `e1071::cmeans()`. `e1071::cmeans()` doesn't have a default value for the number of clusters. Therefore, the `centers` parameter here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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**

Parameters

Id	Type	Default	Levels	Range
centers	untyped	2		-
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	1		-
control	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustCMeans`

Methods

Public methods:

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

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

Usage:

```
LearnerClustCMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("e1071")) {
  learner = mlr3::lrn("clust.cmeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.cobweb`

Cobweb Clustering Learner

Description

A LearnerClust for Cobweb clustering implemented in `RWeka::Cobweb()`. The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

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

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

Meta Information

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

Parameters

Id	Type	Default	Range
A	numeric	1	$[0, \infty)$
C	numeric	0.002	$[0, \infty)$
S	integer	42	$[1, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustCobweb`

Methods

Public methods:

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

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

Usage:

```
LearnerClustCobweb$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("RWeka")) {
  learner = mlr3::lrn("clust.cobweb")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

mlr_learners_clust.dbscan
Density-Based Clustering Learner

Description

A LearnerClust for density-based clustering implemented in `dbscan::dbscan()`. The predict method uses `dbscan::predict.dbscan_fast()` to compute the cluster memberships for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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		$[0, \infty)$
borderPoints	logical	TRUE	TRUE, FALSE	-
weights	untyped	-		-
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` -> `mlr3cluster::LearnerClust` -> `LearnerClustDBSCAN`

Methods

Public methods:

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

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

Usage:

```
LearnerClustDBSCAN$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("dbscan")) {
  learner = mlr3::lrn("clust.dbscan")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.diana`

Divisive Hierarchical Clustering Learner

Description

A LearnerClust for divisive hierarchical clustering implemented in `cluster::diana()`. The predict method uses `stats::cutree()` which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter `k`). The default value for `k` is 2.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	-
trace.lev	integer	0		$[0, \infty)$
k	integer	2		$[1, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustDiana`

Methods

Public methods:

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

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

Usage:

```
LearnerClustDiana$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("cluster")) {
  learner = mlr3::lrn("clust.diana")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

mlr_learners_clust.em *Expectation-Maximization Clustering Learner*

Description

A LearnerClust for Expectation-Maximization clustering implemented in `RWeka::list_Weka_interfaces()`. The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

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

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

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- 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	-
output_debug_info	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustEM`

Methods

Public methods:

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

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

Usage:

```
LearnerClustEM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("RWeka")) {
  learner = mlr3::lrn("clust.em")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_clust.fanny — *Fuzzy Analysis Clustering Learner*

Description

A LearnerClust for fuzzy clustering implemented in `cluster::fanny()`. `cluster::fanny()` doesn't have a default value for the number of clusters. Therefore, the `k` parameter which corresponds to the number of clusters here is set to 2 by default. The predict method copies cluster assignments and memberships generated for train data. The predict does not work for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	2		$[1, \infty)$
memb.exp	numeric	2		$[1, \infty)$
metric	character	euclidean	euclidean, manhattan, SqEuclidean	-
stand	logical	FALSE	TRUE, FALSE	-
maxit	integer	500		$[0, \infty)$
tol	numeric	1e-15		$[0, \infty)$
trace.lev	integer	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustFanny`

Methods

Public methods:

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

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

Usage:

```
LearnerClustFanny$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("cluster")) {
  learner = mlr3::lrn("clust.fanny")
  print(learner)
  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.featureless` *Featureless Clustering Learner*

Description

A simple LearnerClust which randomly (but evenly) assigns observations to `num_clusters` partitions (default: 1 partition).

Dictionary

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

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

Meta Information

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

Parameters

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

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustFeatureless`

Methods

Public methods:

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

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

Usage:

```
LearnerClustFeatureless$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("mlr3")) {
  learner = mlr3::lrn("clust.featureless")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.FF` *Farthest First Clustering Learner*

Description

A LearnerClust for Farthest First clustering implemented in `RWeka::FarthestFirst()`. The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

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

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

Meta Information

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

Parameters

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

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustFF`

Methods

Public methods:

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

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

Usage:

```
LearnerClustFarthestFirst$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("RWeka")) {
  learner = mlr3::lrn("clust.ff")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.hclust` *Agglomerative Hierarchical Clustering Learner*

Description

A LearnerClust for agglomerative hierarchical clustering implemented in `stats::hclust()`. Difference Calculation is done by `stats::dist()`

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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			-
distmethod	character	euclidean	euclidean, maximum, manhattan, canberra, binary, minkowski	-
diag	logical	FALSE	TRUE, FALSE	-
upper	logical	FALSE	TRUE, FALSE	-
p	numeric	2		$(-\infty, \infty)$
k	integer	2		$[1, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustHclust`

Methods

Public methods:

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

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

Usage:

`LearnerClustHclust$new()`

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

Usage:

`LearnerClustHclust$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("stats")) {
  learner = mlr3::lrn("clust.hclust")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.kkmeans`
Kernel K-Means Clustering Learner

Description

A LearnerClust for kernel k-means clustering implemented in `kernlab::kkmeans()`. `kernlab::kkmeans()` doesn't have a default value for the number of clusters. Therefore, the `centers` parameter here is set to 2 by default. Kernel parameters have to be passed directly and not by using the `kpar` list in `kkmeans`. The predict method finds the nearest center in kernel distance to assign clusters for new data points.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	2		-
kernel	character	rbfdot	vanilladot, polydot, rbfdot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		$[0, \infty)$
degree	integer	3		$[1, \infty)$
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` -> `mlr3cluster::LearnerClust` -> `LearnerClustKKMeans`

Methods

Public methods:

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

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

Usage:

```
LearnerClustKKMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("kernlab")) {
  learner = mlr3::lrn("clust.kkmeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_clust.kmeans

K-Means Clustering Learner

Description

A LearnerClust for k-means clustering implemented in `stats::kmeans()`. `stats::kmeans()` doesn't have a default value for the number of clusters. Therefore, the `centers` parameter here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	2		-
iter.max	integer	10		$[1, \infty)$
algorithm	character	Hartigan-Wong	Hartigan-Wong, Lloyd, Forgy, MacQueen	-
nstart	integer	1		$[1, \infty)$
trace	integer	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustKMeans`

Methods

Public methods:

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

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

Usage:

```
LearnerClustKMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("stats") && requireNamespace("clue")) {
  learner = mlr3::lrn("clust.kmeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.meanshift`

Mean Shift Clustering Learner

Description

A LearnerClust for Mean Shift clustering implemented in `LPCM::ms()`. There is no predict method for `LPCM::ms()`, so the method returns cluster labels for the 'training' data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	Range
h	untyped	-	-
subset	untyped	-	-
scaled	integer	1	$[0, \infty)$
iter	integer	200	$[1, \infty)$
thr	numeric	0.01	$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustMeanShift`

Methods

Public methods:

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

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

Usage:

```
LearnerClustMeanShift$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("LPCM")) {
  learner = mlr3::lrn("clust.meanshift")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

mlr_learners_clust.MiniBatchKMeans

Mini Batch K-Means Clustering Learner

Description

A LearnerClust for mini batch k-means clustering implemented in `ClusterR::MiniBatchKmeans()`. `ClusterR::MiniBatchKmeans()` doesn't have a default value for the number of clusters. Therefore, the `clusters` parameter here is set to 2 by default. The predict method uses `ClusterR::predict_MBatchKMeans()` to compute the cluster memberships for new data. The learner supports both partitional and fuzzy clustering.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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	2		$[1, \infty)$
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			-
tol	numeric	1e-04		$[0, \infty)$
tol_optimal_init	numeric	0.3		$[0, \infty)$
seed	integer	1		$(-\infty, \infty)$

Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustMiniBatchKMeans

Methods

Public methods:

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

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

Usage:

```
LearnerClustMiniBatchKMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("ClusterR")) {
  learner = mlr3::lrn("clust.MBatchKMeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_clust.pam

Partitioning Around Medoids Clustering Learner

Description

A LearnerClust for PAM clustering implemented in `cluster::pam()`. `cluster::pam()` doesn't have a default value for the number of clusters. Therefore, the `k` parameter which corresponds to the number of clusters here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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**

Parameters

Id	Type	Default	Levels	Range
k	integer	2		$[1, \infty)$
metric	character	-	euclidian, manhattan	-
medoids	untyped			-
stand	logical	FALSE	TRUE, FALSE	-
do.swap	logical	TRUE	TRUE, FALSE	-
pamonce	integer	0		$[0, 5]$
trace.lev	integer	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustPAM`

Methods

Public methods:

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

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

Usage:

```
LearnerClustPAM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("cluster")) {
  learner = mlr3::lrn("clust.pam")
  print(learner)
  # available parameters:
  learner$param_set$ids()
}
```

mlr_learners_clust.SimpleKMeans

K-Means Clustering Learner from Weka

Description

A LearnerClust for Simple K Means clustering implemented in `RWeka::SimpleKMeans()`. The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

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

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

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- 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	100		$[1, \infty)$
init	integer	0		$[0, 3]$
M	logical	FALSE	TRUE, FALSE	-
max_candidates	integer	100		$[1, \infty)$
min_density	integer	2		$[1, \infty)$
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	-1.5		$(-\infty, \infty)$
V	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustSimpleKMeans`

Methods

Public methods:

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

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

Usage:

```
LearnerClustSimpleKMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("RWeka")) {
  learner = mlr3::lrn("clust.SimpleKMeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_clust.xmeans` *X-means Clustering Learner*

Description

A LearnerClust for X-means clustering implemented in `RWeka::XMeans()`. The predict method uses `RWeka::predict.Weka_clusterer()` to compute the cluster memberships for new data.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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		$[0, \infty)$
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		$[1, \infty)$
U	integer	0		$[0, \infty)$
use_kdtree	logical	FALSE	TRUE, FALSE	-
N	untyped	-		-
O	untyped	-		-
Y	untyped	-		-
output_debug_info	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustXMeans`

Methods

Public methods:

- `LearnerClustXMeans$new()`
- `LearnerClustXMeans$clone()`

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

Usage:

```
LearnerClustXMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (requireNamespace("RWeka")) {
  learner = mlr3::lrn("clust.xmeans")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_measures_clust.ch` *Calinski Harabasz Pseudo F-Statistic*

Description

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Calinski_Harabasz".

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Calinski_Harabasz".

Format

`R6::R6Class()` inheriting from MeasureClust.

`R6::R6Class()` inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.ch")
msr("clust.ch")
```

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.ch")
msr("clust.ch")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required predict type: `partition`

- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required predict type: `partition`

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.

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.db`, `mlr_measures_clust.dunn`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.wss`

Other cluster measures: `mlr_measures_clust.db`, `mlr_measures_clust.dunn`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.wss`

`mlr_measures_clust.db` *Davies-Bouldin Cluster Separation Measure*

Description

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Davies_Bouldin".

Format

`R6::R6Class()` inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.db")
msr("clust.db")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required predict type: `partition`

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.ch`, `mlr_measures_clust.dunn`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.wss`

`mlr_measures_clust.dunn` *Dunn Index*

Description

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Dunn".

Format

`R6::R6Class()` inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.dunn")
msr("clust.dunn")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required predict type: `partition`

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.ch`, `mlr_measures_clust.db`, `mlr_measures_clust.silhouette`, `mlr_measures_clust.wss`

mlr_measures_clust.silhouette

Rousseeuw's Silhouette Quality Index

Description

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Silhouette".

Format

`R6::R6Class()` inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.silhouette")
msr("clust.silhouette")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required predict type: `partition`

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.ch`, `mlr_measures_clust.db`, `mlr_measures_clust.dunn`, `mlr_measures_clust.wss`

mlr_measures_clust.wss

Within Sum of Squares

Description

The score function calls `clusterCrit::intCriteria()` from package **clusterCrit**. Argument `crit` is set to "Trace_W".

Format

`R6::R6Class()` inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

```
mlr_measures$get("clust.wss")
msr("clust.wss")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required predict type: `partition`

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.ch`, `mlr_measures_clust.db`, `mlr_measures_clust.dunn`, `mlr_measures_clust.silhouette`

`mlr_tasks_usarrests` *US Arrests Cluster Task*

Description

A cluster task for the datasets::USArrests data set.

Format

R6::R6Class inheriting from TaskClust.

Construction

```
mlr_tasks$get("usarrests")
tsk("usarrests")
```

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()`)
  Access to the stored probabilities.

Methods

Public methods:

- `PredictionClust$new()`
- `PredictionClust$clone()`

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

Usage:

```
PredictionClust$new(
  task = NULL,
  row_ids = task$row_ids,
  partition = NULL,
  prob = NULL,
  check = TRUE
)
```

Arguments:

`task` (TaskClust)
  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()`)
  Vector of cluster partitions.

`prob` (`matrix()`)
  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 using `max.col()` with `ties.method` set to `"first"`.

`check (logical(1))`
If `TRUE`, performs some argument checks and predict type conversions.

Method `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.kmeans")
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 mlr_tasks.

Super classes

`mlr3::Task` -> `mlr3::TaskUnsupervised` -> `TaskClust`

Methods

Public methods:

- `TaskClust$new()`
- `TaskClust$clone()`

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

Usage:

```
TaskClust$new(id, backend, label = NA_character_)
```

Arguments:

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

`backend` (DataBackend)
: Either a DataBackend, or any object which is convertible to a DataBackend with `as_data_backend()`. E.g., a `data.frame()` will be converted to a DataBackendDataTable.

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

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(mlr3)
library(mlr3cluster)
task = TaskClust$new("usarrests", backend = USArrests)
task$task_type

# possible properties:
mlr_reflections$task_properties$clust
```

Index