# Package: mlr3cluster (via r-universe)

August 29, 2024

**Title** Cluster Extension for 'mlr3'

**Version** 0.1.9

**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, fpc, cluster, data.table, mlr3misc (>= 0.10.0), paradox (>= 0.10.0), R6, stats

**Suggests** dbscan, e1071, ClusterR, kernlab, apcluster, LPCM, mclust, mlbench, RWeka, testthat (>= 3.0.0)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE, r6 = TRUE)

**RoxygenNote** 7.3.1

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

**Config/testthat/edition** 3

**Repository** https://mlr-org.r-universe.dev

**RemoteUrl** https://github.com/mlr-org/mlr3cluster

**RemoteRef** v0.1.9

**RemoteSha** 9fcba0d5aeb4459657d85de0f70d3914746d9d9e

# Contents

## 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)<br>Object to convert. |
| ... | (any)<br>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)<br>Object to convert. |
| ... | (any)<br>Additional arguments. |
| clone | (logical(1))<br>If TRUE, ensures that the returned object is not the same as the input x. |
| id | (character(1))<br>Id for the new task. Defaults to the (deparsed and substituted) name of the data argument. |
| data | (data.frame())<br>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_,
  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 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.

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().

**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:
learner = lrn("clust.kmeans")
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()`

**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.

## 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: 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.
- 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.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.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean` `mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

### 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. 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 | - | | - |
| 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.

## 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: 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.
- 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.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.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmean
mlr_learners_clust.kmeans, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## 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.

## 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: 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.
- 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.cobweb, mlr_learners_clust.dbscan, mlr_learners_clust.dbscan_
mlr_learners_clust.diana, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.featurele
mlr_learners_clust.ff, mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmean
mlr_learners_clust.kmeans, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## 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.

### 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: 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.
- 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.cmeans`, `mlr_learners_clust.dbscan`, `mlr_learners_clust.dbscan_`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

## Examples

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

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

---

# mlr_learners_clust.dbscan

*Density-based Spatial Clustering of Applications with Noise (DBSCAN) Clustering Learner*

---

## Description

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

## 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.

### 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: 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.
- 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.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan_`
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

## Examples

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

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

---

`mlr_learners_clust.dbscan_fpc`

*Density-based Spatial Clustering of Applications with Noise (DBSCAN) Clustering Learner*

---

## Description

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

## Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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, raw, dist | - |
| seeds | logical | TRUE | TRUE, FALSE | - |
| showplot | untyped | FALSE | | - |
| countmode | untyped | | | - |

### Super classes

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

### Methods

#### Public methods:

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

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

*Usage:*

```
LearnerClustDBSCANfpc$new()
```

**Method** `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: 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.
- 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.cmeans`, `mlr_learners_clust.cobweb`, `mlr_learners_clust.dbscan`,
`mlr_learners_clust.diana`, `mlr_learners_clust.em`, `mlr_learners_clust.fanny`, `mlr_learners_clust.featurele`
`mlr_learners_clust.ff`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

## Examples

```
if (requireNamespace("fpc")) {
  learner = mlr3::lrn("clust.dbscan_fpc")
  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.

### 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: 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.
- 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.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan,
mlr_learners_clust.dbscan_fpc, mlr_learners_clust.em, mlr_learners_clust.fanny, mlr_learners_clust.feat
mlr_learners_clust.ff, mlr_learners_clust.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmean
mlr_learners_clust.kmeans, mlr_learners_clust.mclust, mlr_learners_clust.meanshift,
mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## 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.

## 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: 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.
- 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.cmeans, mlr_learners_clust.cobweb, mlr_learners_clust.dbscan,
mlr_learners_clust.dbscan_fpc, mlr_learners_clust.diana, mlr_learners_clust.fanny,
mlr_learners_clust.featureless, mlr_learners_clust.ff, mlr_learners_clust.hclust,
mlr_learners_clust.hdbscan, mlr_learners_clust.kkmeans, mlr_learners_clust.kmeans,
mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.optics,
mlr_learners_clust.pam, mlr_learners_clust.xmeans

## 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.

## 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: 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.
- 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.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.feat`
`mlr_learners_clust.ff`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

## 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.

## 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: 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.
- 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.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.fann`
`mlr_learners_clust.ff`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmean`
`mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`,
`mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xmeans`

## 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.

## 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: 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.
- 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.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.fanr`
`mlr_learners_clust.featureless`, `mlr_learners_clust.hclust`, `mlr_learners_clust.hdbscan`,
`mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`,
`mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xme`

## 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.

## 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: 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.
- 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.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.fanr
mlr_learners_clust.featureless, mlr_learners_clust.ff, mlr_learners_clust.hdbscan,
mlr_learners_clust.kkmeans, mlr_learners_clust.kmeans, mlr_learners_clust.mclust,
mlr_learners_clust.meanshift, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xme

## Examples

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

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

---

`mlr_learners_clust.hdbscan`
*Hierarchical DBSCAN (HDBSCAN) Clustering Learner*

---

## Description

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

## Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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 | - | | $[0, \infty)$ |
| gen_hdbscan_tree | logical | FALSE | TRUE, FALSE | - |
| gen_simplified_tree | logical | FALSE | TRUE, FALSE | - |
| verbose | logical | FALSE | TRUE, FALSE | - |

## Super classes

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

## Methods

### Public methods:

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

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

*Usage:*

```
LearnerClustHDBSCAN$new()
```

**Method** `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: 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.
- 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.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.fann`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`, `mlr_learners_clust.mclust`,
`mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xme`

## Examples

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

## 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: 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.
- 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.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.fanr
mlr_learners_clust.featureless, mlr_learners_clust.ff, mlr_learners_clust.hclust,
mlr_learners_clust.hdbscan, mlr_learners_clust.kmeans, mlr_learners_clust.mclust,
mlr_learners_clust.meanshift, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xme

## 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.

## 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: 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.
- 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.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.fanr`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.mclust`,
`mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xme`

## Examples

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

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

---

`mlr_learners_clust.MBatchKMeans` *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.

## 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: 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.
- 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.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.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmean mlr_learners_clust.kmeans, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## Examples

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

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

---

# mlr_learners_clust.mclust

*Gaussian Mixture Models-Based Clustering Learner*

---

## Description

A LearnerClust for model-based clustering implemented in mclust::Mclust(). The predict method uses mclust::predict.Mclust() 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.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 |
|---|---|---|
| G | untyped | :, 1, 9 |
| modelNames | untyped | - |
| prior | untyped | - |
| control | untyped | mclust::emControl |
| initialization | untyped | - |
| x | untyped | - |

## Super classes

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

## Methods

### Public methods:

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

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

*Usage:*

```
LearnerClustMclust$new()
```

**Method** `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: 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.
- 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.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.fanr`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`,
`mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`, `mlr_learners_clust.pam`, `mlr_learners_clust.xme`

## Examples

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

### 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: 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.
- 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.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.fanr
mlr_learners_clust.featureless, mlr_learners_clust.ff, mlr_learners_clust.hclust,
mlr_learners_clust.hdbscan, mlr_learners_clust.kkmeans, mlr_learners_clust.kmeans,
mlr_learners_clust.mclust, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## Examples

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

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

---

# mlr_learners_clust.optics

*Ordering Points to Identify the Clustering Structure (OPTICS) Clustering Learner*

---

## Description

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

## Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `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 | NULL | | $[0, \infty)$ |
| minPts | integer | 5 | | $[0, \infty)$ |
| 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 | | $(-\infty, \infty)$ |
| eps_cl | numeric | - | | $[0, \infty)$ |

### Super classes

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

### Methods

#### Public methods:

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

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

*Usage:*

```
LearnerClustOPTICS$new()
```

**Method** `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: 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.
- 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.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.fann`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`,
`mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.pam`, `mlr_learners_clust.xme`

## Examples

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

## 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: 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.
- 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.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.fanr`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`,
`mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`,
`mlr_learners_clust.xmeans`

## 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.

## 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: 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.

- 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.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.hclust, mlr_learners_clust.hdbscan, mlr_learners_clust.kkmean mlr_learners_clust.kmeans, mlr_learners_clust.mclust, mlr_learners_clust.meanshift, mlr_learners_clust.optics, mlr_learners_clust.pam, mlr_learners_clust.xmeans

## 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.

### 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: 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.
- 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.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.fanr`
`mlr_learners_clust.featureless`, `mlr_learners_clust.ff`, `mlr_learners_clust.hclust`,
`mlr_learners_clust.hdbscan`, `mlr_learners_clust.kkmeans`, `mlr_learners_clust.kmeans`,
`mlr_learners_clust.mclust`, `mlr_learners_clust.meanshift`, `mlr_learners_clust.optics`,
`mlr_learners_clust.pam`

## 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 `fpc::cluster.stats()` from package **fpc**. "ch" is used subset output of the function call.

## Format

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

### Construction

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

### 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.dunn, mlr_measures_clust.silhouette, mlr_measures_clust.wss

---

## mlr_measures_clust.dunn

*Dunn Index*

---

### Description

The score function calls fpc::cluster.stats() from package **fpc**. "dunn" is used subset output of the function call.

### 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.silhouette`, `mlr_measures_clust.wss`

---

`mlr_measures_clust.silhouette`

*Rousseeuw's Silhouette Quality Index*

---

## Description

The score function calls `cluster::silhouette()` from package **cluster**. "sil_width" is used subset output of the function call.

## 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.dunn`, `mlr_measures_clust.wss`

## mlr_measures_clust.wss *Within Sum of Squares*

### Description

The score function calls `fpc::cluster.stats()` from package **fpc**. "within.cluster.ss" is used subset output of the function call.

### 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.dunn`, `mlr_measures_clust.silhouette`

## mlr_tasks_ruspini *Ruspini Cluster Task*

### Description

A cluster task for the cluster::ruspini data set.

### Format

R6::R6Class inheriting from TaskClust.

### Construction

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

### Source

Ruspini EH (1970). "Numerical methods for fuzzy clustering." *Information Sciences*, **2**(3), 319-350. doi:10.1016/S00200255(70)800561.

---

## mlr_tasks_usarrests *US Arrests Cluster Task*

---

### Description

A cluster task for the datasets::USArrests data set. Rownames are stored as variable `"states"` with column role `"name"`.

### 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