Package 'mlr3cluster'

October 1, 2020

Title Cluster Extension for 'mlr3'

Version 0.1.0

Description Extends the 'mlr3' package with cluster analysis.

License LGPL-3

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

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

Depends R (>= 3.1.0)

Imports backports (>= 1.1.10), checkmate, clue, clusterCrit, data.table, mlr3 (>= 0.5.0), mlr3misc (>= 0.4.0), paradox, R6

Suggests bibtex, dbscan, e1071, mlbench, RWeka, testthat

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'LearnerClust.R' 'LearnerClustAgnes.R' 'LearnerClustCMeans.R' 'LearnerClustDBSCAN.R' 'LearnerClustDiana.R' 'LearnerClustFanny.R' 'LearnerClustFeatureless.R' 'LearnerClustKMeans.R' 'LearnerClustPAM.R' 'LearnerClustXMeans.R' 'MeasureClust.R' 'measures.R' 'MeasureClustInternal.R' 'PredictionClust.R' 'PredictionDataClust.R' 'TaskClust.R' 'TaskClust_usarrest.R' 'helper.R' 'zzz.R'

NeedsCompilation no

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

Maintainer Damir Pulatov <dpulatov@uwyo.edu>

Repository CRAN

Date/Publication 2020-10-01 09:10:02 UTC

R topics documented:

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

Description

Extends the 'mlr3' package with cluster analysis.

Author(s)

Maintainer: Damir Pulatov <dpulatov@uwyo.edu>

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

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`

Methods

Public methods:

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

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

Usage:

```
LearnerClust$new(
  id,
  param_set = ParamSet$new(),
  predict_types = "partition",
  feature_types = character(),
  properties = character(),
  packages = 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()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

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

`MeasureClust` *Cluster Measure*

Description

This measure specializes mlr3::Measure for cluster analysis:

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

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

Super class

`mlr3::Measure` -> `MeasureClust`

Methods

Public methods:

- `MeasureClust$new()`

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

Usage:

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

`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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustAgnes$new()
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.agnes")
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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustCMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.cmeans")
print(learner)

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

mlr_learners_clust.dbscan

Density-Based Clustering Learner

Description

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

Dictionary

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

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

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustDBSCAN$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

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

`mlr_learners_clust.diana`

Divisive Hierarchical Clustering Learner

Description

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

Dictionary

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

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

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustDiana$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.diana")
print(learner)

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

`mlr_learners_clust.fanny` *Fuzzy Analysis Cluster Learner*

Description

A LearnerClust for fuzzy clustering implemented in `cluster::fanny()`. `cluster::fanny()` doesn't have a deafult value for the number of clusters. Therefore, the `k` parameter which correponds 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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustFanny$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
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 assigns first n observations to cluster 1, second n observations to cluster 2, and so on. Hyperparameter `num_clusters` controls the number of clusters and is set to 1 by default. The train method tries to assign cluster memberships to each observation such that each cluster has an equal amount of observations. The predict method uses does the same thing as the train but 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.featureless")
lrn("clust.featureless")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustFeatureless$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.kmeans")
print(learner)

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

`mlr_learners_clust.kmeans`
KMeans Cluster 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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustKMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.kmeans")
print(learner)

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

`mlr_learners_clust.pam`

Partitioning Around Medoids Cluster 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 correponds 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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustPAM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.pam")
print(learner)

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

`mlr_learners_clust.xmeans` *X-means Cluster 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")
```

Super classes

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

Methods

Public methods:

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

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

Usage:

```
LearnerClustXMeans$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
learner = mlr3::lrn("clust.xmeans")
print(learner)

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

mlr_measures_clust.ch *Calinski Harabasz Pseudo F-Statistic*

Description

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

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

mlr_measures_clust.db *Davies-Bouldin Cluster Separation Measure*

Description

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

Format

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

Construction

This measures can be retrieved from the dictionary mlr_measures:

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

Meta Information

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

See Also

Dictionary of Measures: mlr3::mlr_measures

as.data.table(mlr_measures) for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: mlr_measures_clust.ch, mlr_measures_clust.dunn, mlr_measures_clust.silhouette

---

mlr_measures_clust.dunn

Dunn Index

Description

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

Format

R6::R6Class() inheriting from MeasureClust.

Construction

This measures can be retrieved from the dictionary mlr_measures:

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

Meta Information

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

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.ch`, `mlr_measures_clust.db`, `mlr_measures_clust.silhouette`

`mlr_measures_clust.silhouette`

Rousseeuw's Silhouette Quality Index

Description

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

Format

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

Construction

This measures can be retrieved from the dictionary mlr_measures:

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

Meta Information

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

See Also

Dictionary of Measures: mlr3::mlr_measures

`as.data.table(mlr_measures)` for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: `mlr_measures_clust.ch`, `mlr_measures_clust.db`, `mlr_measures_clust.dunn`

mlr_tasks_usarrests *US Arrests Cluster Task*

Description

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

Format

R6::R6Class inheriting from TaskClust.

Construction

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

PredictionClust *Prediction Object for Cluster Analysis*

Description

This object wraps the predictions returned by a learner of class LearnerClust, i.e. the predicted partition and cluster probability.

Super class

`mlr3::Prediction` -> `PredictionClust`

Active bindings

`partition (integer())`
Access the stored partition.

`prob (matrix())`
Access to the stored probabilities.

Methods

Public methods:

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

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

Usage:

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

Arguments:

`task` (TaskClust)
: Task, used to extract defaults for `row_ids`.

`row_ids` (`integer()`)
: Row ids of the predicted observations, i.e. the row ids of the test set.

`partition` (`integer()`)
: Vector of cluster partitions.

`prob` (`matrix()`)
: Numeric matrix of cluster membership probabilities with one column for each cluster and one row for each observation. Columns must be named with cluster numbers, row names are automatically removed. If `prob` is provided, but `partition` is not, the cluster memberships are calculated from the probabilities using `max.col()` with `ties.method` set to `"first"`.

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

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(mlr3)
library(mlr3cluster)
task = tsk("usarrests")
learner = lrn("clust.kmeans")
p = learner$train(task)$predict(task)
p$predict_types
head(as.data.table(p))
```

`TaskClust` *Cluster Task*

Description

This task specializes mlr3::Task for cluster problems. As an unsupervised task, this task has no target column. The `task_type` is set to `"clust"`.

Predefined tasks are stored in the dictionary mlr_tasks.

Super class

`mlr3::Task` -> `TaskClust`

Methods

Public methods:

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

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

Usage:

```
TaskClust$new(id, backend)
```

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.

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