Package ‘mlr3oml’

January 28, 2026

Title Connector Between 'mlr3' and 'OpenML'

Version 0.12.0

Description Provides an interface to 'OpenML.org' to list and download machine learning data, tasks and experiments. The 'OpenML' objects can be automatically converted to 'mlr3' objects. For a more sophisticated interface with more upload options, see the 'OpenML' package.

License LGPL-3

URL https://mlr3oml.mlr-org.com, https://github.com/mlr-org/mlr3oml

BugReports https://github.com/mlr-org/mlr3oml/issues

Depends R (>= 3.1.0)

Imports backports (>= 1.1.6), bit64, checkmate, curl, data.table, jsonlite, lgr, methods, mlr3 (>= 0.16.0), mlr3misc (>= 0.7.0), paradox, R6, stringi, uuid, withr

Suggests DBI, duckdb (>= 0.6.0), mlr3db (>= 0.5.0), qs2, RWeka, testthat (>= 3.0.0), xml2, httr

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.3.3

Config/Needs/website rmarkdown

Author Michel Lang [aut] (ORCID: <https://orcid.org/0000-0001-9754-0393>), Sebastian Fischer [cre, aut] (ORCID: <https://orcid.org/0000-0002-9609-3197>)

Maintainer Sebastian Fischer <sebf.fischer@gmail.com>

Repository CRAN

Date/Publication 2026-01-28 10:00:02 UTC

Contents

mlr3oml-package 2
as_learner.OMLFlow 4
list_oml_collections 4
ocl 8
odt 8
oflw 9
oml_collection 9
oml_data 11
oml_flow 15
oml_object 16
oml_run 18
oml_task 20
orn 22
otsk 23
publish_collection 23
publish_data 24
publish_task 25
read_arff 26
write_arff 27

Index **28**

`mlr3oml-package` *mlr3oml: Connector Between 'mlr3' and 'OpenML'*

Description

Provides an interface to 'OpenML.org' to list and download machine learning data, tasks and experiments. The 'OpenML' objects can be automatically converted to 'mlr3' objects. For a more sophisticated interface with more upload options, see the 'OpenML' package.

Documentation

Start by reading the Large-Scale Benchmarking chapter from the mlr3book.

mlr3 Integration

This package adds the mlr3::Task `"oml"` and the mlr3::Resampling `"oml"` to mlr3::mlr_tasks and mlr3::mlr_resamplings, respectively. For the former you may pass either a `data_id` or a `task_id`, the latter requires a `task_id`. Furthermore it allows to convert the OpenML objects to mlr3 objects using the usual S3 generics such as mlr3::as_task, mlr3::as_learner, mlr3::as_resampling, mlr3::as_resample_result, mlr3::as_benchmark_result or mlr3::as_data_backend. This allows for a frictionless integration of OpenML and mlr3.

Options

- `mlr3oml.cache`: Enables or disables caching globally. If set to `FALSE`, caching is disabled. If set to `TRUE`, cache directory as reported by `backports::R_user_dir()` is used. Alternatively, you can specify a path on the local file system here. Default is `FALSE`.
- `mlr3oml.api_key`: API key to use. All operations supported by this package work without an API key, but you might get rate limited without an API key. If not set, defaults to the value of the environment variable `OPENMLAPIKEY`.
- `mlr3oml.arff_parser`: ARFF parser to use, defaults to the internal one relies on `data.table::fread()`. Can also be set to `"RWeka"` for the parser in **RWeka**.
- `mlr3oml.parquet`: Enables or disables parquet as the default file format. If set to `TRUE`, the parquet version of datasets will be used by default. If set to `FALSE`, the arff version of datasets will be used by default. Note that the OpenML sever is still transitioning from arff to parquet and some features will work better with arff. Default is `FALSE`.
- `mlr3oml.retries`: An integer defining number of retries when downloading data from OpenML. If it is `NULL`, the number of retries is set to 3.

Relevant for developers

- `mlr3oml.test_server`: The default value for whether to use the OpenML test server. Default is `FALSE`.
- `mlr3oml.test_api_key`: API key to use for the test server. If not set, defaults to the value of the environment variable `TESTOPENMLAPIKEY`.

Logging

The **lgr** package is used for logging. To change the threshold, use `lgr::get_logger("mlr3oml")$set_threshold()`.

Author(s)

Maintainer: Sebastian Fischer <`sebf.fischer@gmail.com`> (ORCID)

Authors:

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

See Also

Useful links:

- `https://mlr3oml.mlr-org.com`
- `https://github.com/mlr-org/mlr3oml`
- Report bugs at `https://github.com/mlr-org/mlr3oml/issues`

as_learner.OMLFlow *Convert an OpenML Flow to a mlr3 Learner*

Description

By default this function creates a Pseudo-Learner (that cannot be used for training or prediction) for the given task type. This enables the conversion of OpenML Runs to mlr3::ResampleResults. This is well defined because each subcomponent (i.e. id) can only appear once in a Flow according to the OpenML docs.

Usage

```
## S3 method for class 'OMLFlow'
as_learner(x, task_type = NULL, ...)
```

Arguments

`x`	(OMLFlow) The OMLFlow that is converted to a mlr3::Learner.
`task_type`	(`character(1)`) The task type to constrct a pseudo-learner. For more information see OMLFlow.
`...`	Additional arguments.

list_oml_collections *List Data from OpenML*

Description

This function allows to query data sets, tasks, flows, setups, runs, and evaluation measures from `https://www.openml.org/search?type=data&sort=runs&status=active` using some simple filter criteria.

To find datasets for a specific task type, use `list_oml_tasks()` which supports filtering according to the task type. Another heuristic to search for possible regression tasks is to search for data sets with 0 number of classes, i.e. by specifying `number_classes = 0`.

Usage

```
list_oml_collections(
  uploader = NULL,
  status = "all",
  main_entity_type = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)
```

```
list_oml_data(
  data_id = NULL,
  data_name = NULL,
  number_instances = NULL,
  number_features = NULL,
  number_classes = NULL,
  number_missing_values = NULL,
  tag = NULL,
  uploader = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)

list_oml_evaluations(
  run_id = NULL,
  task_id = NULL,
  measures = NULL,
  tag = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)

list_oml_flows(
  uploader = NULL,
  tag = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)

list_oml_measures(test_server = test_server_default())

list_oml_runs(
  run_id = NULL,
  task_id = NULL,
  tag = NULL,
  flow_id = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)

list_oml_setups(
  flow_id = NULL,
  setup_id = NULL,
  tag = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  ...
)

list_oml_tasks(
  task_id = NULL,
  data_id = NULL,
  number_instances = NULL,
  number_features = NULL,
  number_classes = NULL,
  number_missing_values = NULL,
  tag = NULL,
  limit = limit_default(),
  test_server = test_server_default(),
  type = NULL,
  ...
)
```

Arguments

`uploader`	`(integer(1))` Filter for uploader.
`status`	`(character(1))` Should be one of "active", "in_preparation", "deactivated", "all". By default "all" studies are returned.
`main_entity_type`	`(character(1)` | `NULL)` Filter for main entity type. Can be "run" or "task".
`limit`	`(integer())` Limit the results to `limit` records. Default is the value of option `"mlr3oml.limit"`, defaulting to 5000.
`test_server`	`(character(1))` Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.
`...`	(any) Additional (unsupported) filters, as named arguments.
`data_id`	`(integer())` Vector of data ids to restrict to.
`data_name`	`(character(1))` Filter for name of data set.
`number_instances`	`(integer())` Filter for number of instances.
`number_features`	`(integer())` Filter for number of features.

`number_classes`	(`integer()`) Filter for number of labels of the target (only classification tasks).
`number_missing_values`	(`integer()`) Filter for number of missing values.
`tag`	(`character()`) Filter for tags. You can provide multiple tags as character vector.
`run_id`	(`integer()`) Vector of run ids to restrict to.
`task_id`	(`integer()`) Vector of task ids to restrict to.
`measures`	(`character()`) Vector of evaluation measures to restrict to.
`flow_id`	(`integer(1)`) Filter for flow id.
`setup_id`	(`integer()`) Vector of setup ids to restrict to.
`type`	(`character(1)`) The task type, supported values are: `"clasisf"`, `"regr"`, `"surv"` and `"clust"`.

Details

Filter values are usually provided as single atomic values (typically integer or character). Provide a numeric vector of length 2 (`c(l, u)`) to find matches in the range $[l, u]$.

Note that only a subset of filters is exposed here. For a more feature-complete package, see **OpenML**. Alternatively, you can pass additional filters via `...` using the names of the official API, c.f. the *REST* tab of `https://www.openml.org/apis`.

Value

(`data.table()`) of results, or a null data.table if no data set matches the filter criteria.

References

Casalicchio G, Bossek J, Lang M, Kirchhoff D, Kerschke P, Hofner B, Seibold H, Vanschoren J, Bischl B (2017). "OpenML: An R Package to Connect to the Machine Learning Platform OpenML." *Computational Statistics*, 1–15. doi:10.1007/s0018001707422.

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
```

```
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

ocl — *Syntactic Sugar for Collection Construction*

Description

Creates an `OMLCollection` instance.

Usage

```
ocl(id, test_server = test_server_default())
```

Arguments

`id`	(`integer(1)`) OpenML id for the object.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Value

(`OMLCollection`)

odt — *Syntactic Sugar for Data Construction*

Description

Creates an `OMLData` instance.

Usage

```
odt(id, parquet = parquet_default(), test_server = test_server_default())
```

Arguments

`id`	(`integer(1)`) OpenML id for the object.
`parquet`	(`logical(1)`) Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Value

(`OMLData`)

`oflw` *Syntactic Sugar for Flow Construction*

Description

Creates an `OMLFlow` instance.

Usage

```
oflw(id, test_server = test_server_default())
```

Arguments

`id`	(`integer(1)`) OpenML id for the object.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Value

(`OMLFlow`)

`oml_collection` *OpenML Collection*

Description

This is the class for collections (previously known as studies) served on `https://www.openml.org`. A collection can either be a task collection or run collection. This object can also be constructed using the sugar function `ocl()`.

Run Collection

A run collection contains runs, flows, datasets and tasks. The primary object are the runs (`main_entity_type` is `"run"`). The the flows, datasets and tasks are those used in the runs.

Task Collection A task collection (`main_entity_type = "task"`) contains tasks and datasets. The primary object are the tasks (`main_entity_type` is `"task"`). The datasets are those used in the tasks.

Note: All Benchmark Suites on OpenML are also collections.

Caching

Because collections on OpenML can be modified (ids can be added), it is not possible to cache this object.

mlr3 Intergration

- Obtain a list of mlr3::Tasks using mlr3::as_tasks.
- Obtain a list of mlr3::Resamplings using mlr3::as_resamplings.
- Obtain a list of mlr3::Learners using mlr3::as_learners (if main_entity_type is "run").
- Obtain a mlr3::BenchmarkResult using mlr3::as_benchmark_result (if main_entity_type is "run").

Super class

`mlr3oml::OMLObject` -> `OMLCollection`

Active bindings

`desc (list())`
: Colllection description (meta information), downloaded and converted from the JSON API response.

`parquet (logical(1))`
: Whether to use parquet.

`main_entity_type (character(n))`
: The main entity type, either `"run"` or `"task"`.

`flow_ids (integer(n))`
: An vector containing the flow ids of the collection.

`data_ids (integer(n))`
: An vector containing the data ids of the collection.

`run_ids (integer(n))`
: An vector containing the run ids of the collection.

`task_ids (integer(n))`
: An vector containing the task ids of the collection.

Methods

Public methods:

- `OMLCollection$new()`
- `OMLCollection$print()`
- `OMLCollection$download()`
- `OMLCollection$clone()`

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

Usage:

```
OMLCollection$new(id, test_server = test_server_default())
```

Arguments:

`id (integer(1))`
OpenML id for the object.

`test_server (character(1))`
Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Method `print()`**:** Prints the object.

Usage:

`OMLCollection$print()`

Method `download()`**:** Downloads the whole object for offline usage.

Usage:

`OMLCollection$download()`

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

Usage:

`OMLCollection$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

`oml_data` *Interface to OpenML Data Sets*

Description

This is the class for data sets served on OpenML. This object can also be constructed using the sugar function `odt()`.

mlr3 Integration

- A mlr3::Task can be obtained by calling `mlr3::as_task()`. The target column must either be the default target (this is the default behaviour) or one of `$feature_names`. In case the target is specified to be one of `$feature_names`, the default target is added to the features of the task.
- A mlr3::DataBackend can be obtained by calling `mlr3::as_data_backend()`. Depending on the selected file-type, the returned backend is a mlr3::DataBackendDataTable (arff) or mlr3db::DataBackendDuckDB (parquet). Note that a converted backend can contain columns beyond the target and the features (id column or ignore columns).

Name conversion

Column names that don't comply with R's naming scheme are renamed (see `base::make.names()`). This means that the names can differ from those on OpenML.

File Format

The datasets stored on OpenML are either stored as (sparse) ARFF or parquet. When creating a new `OMLData` object, the constructor argument `parquet` allows to switch between arff and parquet. Note that not necessarily all data files are available as parquet. The option `mlr3oml.parquet` can be used to set a default. If `parquet` is `TRUE` but not available, `"arff"` will be used as a fallback.

ARFF Files

This package comes with an own reader for ARFF files, based on `data.table::fread()`. For sparse ARFF files and if the **RWeka** package is installed, the reader automatically falls back to the implementation in (`RWeka::read.arff()`).

Parquet Files

For the handling of parquet files, we rely on **duckdb** and **DBI**.

Super class

`mlr3oml::OMLObject` -> `OMLData`

Active bindings

`qualities (data.table())`
: Data set qualities (performance values), downloaded from the JSON API response and converted to a `data.table::data.table()` with columns `"name"` and `"value"`.

`tags (character())`
: Returns all tags of the object.

`parquet (logical(1))`
: Whether to use parquet.

`data (data.table())`
: Returns the data (without the row identifier and ignore id columns).

`features (data.table())`
: Information about data set features (including target), downloaded from the JSON API response and converted to a `data.table::data.table()` with columns:

  - `"index"` (`integer()`): Column position.
  - `"name"` (`character()`): Name of the feature.
  - `"data_type"` (`factor()`): Type of the feature: `"nominal"` or `"numeric"`.
  - `"nominal_value"` (`list()`): Levels of the feature, or `NULL` for numeric features.
  - `"is_target"` (`logical()`): `TRUE` for target column, `FALSE` otherwise.
  - `"is_ignore"` (`logical()`): `TRUE` if this feature should be ignored. Ignored features are removed automatically from the data set.
  - `"is_row_identifier"` (`logical()`): `TRUE` if the column encodes a row identifier. Row identifiers are removed automatically from the data set.
  - `"number_of_missing_values"` (`integer()`): Number of missing values in the column.

`target_names (character())`
: Name of the default target, as extracted from the OpenML data set description.

`feature_names (character())`
: Name of the features, as extracted from the OpenML data set description.

`nrow (integer())`
: Number of observations, as extracted from the OpenML data set qualities.

`ncol (integer())`
: Number of features (including targets), as extracted from the table of data set features. This excludes row identifiers and ignored columns.

`license (character())`
: Returns all license of the dataset.

`parquet_path (character())`
: Downloads the parquet file (or loads from cache) and returns the path of the parquet file. Note that this also normalizes the names of the parquet file.

Methods

Public methods:

- `OMLData$new()`
- `OMLData$print()`
- `OMLData$download()`
- `OMLData$quality()`
- `OMLData$clone()`

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

Usage:

```
OMLData$new(
  id,
  parquet = parquet_default(),
  test_server = test_server_default()
)
```

Arguments:

`id (integer(1))`
: OpenML id for the object.

`parquet (logical(1))`
: Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.

`test_server (character(1))`
: Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Method `print()`**:** Prints the object. For a more detailed printer, convert to a mlr3::Task via `as_task()`.

Usage:

```
OMLData$print()
```

Method `download()`**:** Downloads the whole object for offline usage.

Usage:

```
OMLData$download()
```

Method `quality()`**:** Returns the value of a single OpenML data set quality.

Usage:

```
OMLData$quality(name)
```

Arguments:

`name (character(1))`
: Name of the quality to extract.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

oml_flow *Interface to OpenML Flows*

Description

This is the class for flows served on OpenML. Flows represent machine learning algorithms. This object can also be constructed using the sugar function `oflw()`.

mlr3 Integration

- Obtain a mlr3::Learner using `mlr3::as_learner()`.

Super class

`mlr3oml::OMLObject` -> `OMLFlow`

Active bindings

`parameter` (`data.table`)
  The parameters of the flow.

`dependencies` (`character()`)
  The dependencies of the flow.

`tags` (`character()`)
  Returns all tags of the object.

Methods

Public methods:

- `OMLFlow$new()`
- `OMLFlow$print()`
- `OMLFlow$download()`
- `OMLFlow$clone()`

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

Usage:

```
OMLFlow$new(id, test_server = test_server_default())
```

Arguments:

`id` (`integer(1)`)
  OpenML id for the object.

`test_server` (`character(1)`)
  Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Method `print()`**:** Prints the object.

Usage:

```
OMLFlow$print()
```

Method `download()`: Downloads the whole object for offline usage.

Usage:

```
OMLFlow$download()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

`oml_object` *Abstract Base Class for OpenML objects.*

Description

All OML Objects inherit from this class. Don't use his class directly.

Active bindings

`desc` (`list()`)
: Description of OpenML object.

`cache_dir` (`logical(1)` | `character(1)`)
: Stores the location of the cache for objects retrieved from OpenML. If set to `FALSE`, caching is disabled. Objects from the test server are stored in the subdirectory 'test', those from the public server are stored in the subdirectory 'public'.

  The package **qs2** is required for caching.

`id` (`integer(1)`)
: OpenML data id.

`server` (`character(1)`)
: The server for this object.

`man` (`character(1)`)
: The manual entry.

`name` (`character(1)`)
: The name of the object.

`type` (`character()`)
: The type of OpenML object (e.g. task, run, ...).

`test_server` (`logical(1)`)
: Whether the object is using the test server.

Methods

Public methods:

- `OMLObject$new()`
- `OMLObject$help()`
- `OMLObject$clone()`

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

Usage:

```
OMLObject$new(id, test_server = test_server_default(), type)
```

Arguments:

`id` (`integer(1)`)
: OpenML id for the object.

`test_server` (`character(1)`)
: Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

`type` (`charcater()`)
: The type of OpenML object (e.g. run, task, ...).

Method `help()`**:** Opens the corresponding help page referenced by field `$man`.

Usage:

```
OMLObject$help()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`oml_run` *Interface to OpenML Runs*

Description

This is the class for OpenML Runs, which are conceptually similar to mlr3::ResampleResults. This object can also be constructed using the sugar function `oml_run()`.

OpenML Integration

- A OMLTask is returned by accessing the active field `$task`.
- A OMLData is returned by accessing the active field `$data` (short for `$task$data`)
- A OMLFlow is returned by accessing the active field `$flow`.
- The raw predictions are returned by accessing the active field `$prediction`.

mlr3 Integration

- A mlr3::ResampleResult is returned when calling `mlr3::as_resample_result()`.
- A mlr3::Task is returned when calling `mlr3::as_task()`.
- A mlr3::DataBackend is returned when calling `mlr3::as_data_backend()`.
- A instantiated mlr3::Resampling is returned when calling `mlr3::as_resampling()`.

Super class

`mlr3oml::OMLObject` -> `OMLRun`

Active bindings

`flow_id` (`integer(1)`)
: The id of the flow.

`flow` (OMLFlow)
: The OpenML Flow.

`tags` (`character()`)
: Returns all tags of the object.

`parquet` (`logical(1)`)
: Whether to use parquet.

`task_id` (`character(1)`)
: The id of the task solved by this run.

`task` (OMLTask)
: The task solved by this run.

`data_id` (`integer(1)`)
: The id of the dataset.

`data` (OMLData)
: The data used in this run.

`task_type (character())`
  The task type.

`parameter_setting data.table())`
  The parameter setting for this run.

`prediction (data.table())`
  The raw predictions of the run as returned by OpenML, not in standard mlr3 format. Formatted predictions are accessible after converting to a mlr3::ResampleResult via `as_resample_result()`.

`evaluation (data.table())`
  The evaluations calculated by the OpenML server.

Methods

Public methods:

- `OMLRun$new()`
- `OMLRun$print()`
- `OMLRun$download()`
- `OMLRun$clone()`

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

Usage:

```
OMLRun$new(
  id,
  parquet = parquet_default(),
  test_server = test_server_default()
)
```

Arguments:

`id (integer(1))`
  OpenML id for the object.

`parquet (logical(1))`
  Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.

`test_server (character(1))`
  Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Method `print()`**:** Prints the object.

Usage:

```
OMLRun$print()
```

Method `download()`**:** Downloads the whole object for offline usage.

Usage:

```
OMLRun$download()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

`oml_task` *Interface to OpenML Tasks*

Description

This is the class for tasks served on OpenML. It consists of a dataset and other meta-information such as the target variable for supervised problems. This object can also be constructed using the sugar function `otsk()`.

mlr3 Integration

- Obtain a mlr3::Task by calling `as_task()`.
- Obtain a mlr3::Resampling by calling `as_resampling()`.

Super class

`mlr3oml::OMLObject` -> `OMLTask`

Active bindings

`estimation_procedure` (`list()`)
: The estimation procedure, returns `NULL` if none is available.

`task_splits` (`data.table()`)
: A data.table containing the splits as provided by OpenML.

`tags` (`character()`)
: Returns all tags of the object.

`parquet` (`logical(1)`)
: Whether to use parquet.

`name` (`character(1)`)
: Name of the task, extracted from the task description.

`task_type (character(1))`
  The OpenML task type.

`data_id (integer())`
  Data id, extracted from the task description.

`data (OMLData)`
  Access to the underlying OpenML data set via a OMLData object.

`nrow (integer())`
  Number of rows, extracted from the OMLData object.

`ncol (integer())`
  Number of columns, as extracted from the OMLData object.

`target_names (character())`
  Name of the targets, as extracted from the OpenML task description.

`feature_names (character())`
  Name of the features (without targets of this OMLTask).

`data_name (character())`
  Name of the dataset (inferred from the task name).

Methods

Public methods:

- `OMLTask$new()`
- `OMLTask$print()`
- `OMLTask$download()`
- `OMLTask$clone()`

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

Usage:

```
OMLTask$new(
  id,
  parquet = parquet_default(),
  test_server = test_server_default()
)
```

Arguments:

`id (integer(1))`
  OpenML id for the object.

`parquet (logical(1))`
  Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.

`test_server (character(1))`
  Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Method `print()`**:** Prints the object. For a more detailed printer, convert to a mlr3::Task via `$task`.

Usage:

`OMLTask$print()`

Method `download()`**:** Downloads the whole object for offline usage.

Usage:

`OMLTask$download()`

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

Usage:

`OMLTask$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi:10.1145/2641190.2641198.

Examples

```
# For technical reasons, examples cannot be included in this R package.
# Instead, these are some relevant resources:
#
# Large-Scale Benchmarking chapter in the mlr3book:
# https://mlr3book.mlr-org.com/chapters/chapter11/large-scale_benchmarking.html
#
# Package Article:
# https://mlr3oml.mlr-org.com/articles/tutorial.html
```

`orn` *Syntactic Sugar for Run Construction*

Description

Creates an `OMLRun` instance.

Usage

```
orn(id, parquet = parquet_default(), test_server = test_server_default())
```

Arguments

`id`	(`integer(1)`) OpenML id for the object.
`parquet`	(`logical(1)`) Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Value

(`OMLRun`)

`otsk` *Syntactic Sugar for Task Construction*

Description

Creates an `OMLTask` instance.

Usage

```
otsk(id, parquet = parquet_default(), test_server = test_server_default())
```

Arguments

`id`	(`integer(1)`) OpenML id for the object.
`parquet`	(`logical(1)`) Whether to use parquet instead of arff. If parquet is not available, it will fall back to arff. Defaults to value of option `"mlr3oml.parquet"` or `FALSE` if not set.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

Value

(`OMLTask`)

`publish_collection` *Publish a Collection to OpenML*

Description

Publish a collection to OpenML This can also be achieved through the website.

Usage

```
publish_collection(
  ids,
  name,
  desc,
  main_entity_type = "task",
  alias = NULL,
  api_key = NULL,
  test_server = test_server_default()
)
```

Arguments

`ids`	(`integer()`) The IDs to include in the collection. Depending on the main entity tupe, these can be task or run IDs.
`name`	(`character(1)`) The name for the collection.
`desc`	(`character(1)`) The description of the collection.
`main_entity_type`	(`character(1)`) The main entity type of the collection. Can be either "task" or "run".
`alias`	(`character(1)`) The alias for the collection.
`api_key`	(`character(1)`) The API key to perform the action, if left NULL it first tries the "mlr3oml.api_key" R option and then the environment variable `OPENMLAPIKEY`. In case `test_server` is TRUE (only relevant for developers) the test server API key is used, i.e. first the option "mlr3oml.test_api_key" and then the environment variable `TESTOPENMLAPIKEY`.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

`publish_data` *Upload data to OpenML*

Description

Upload a dataset to OpenML. This can also be achieved through the website.

Usage

```
publish_data(
  data,
  name,
  desc,
  license = NULL,
  default_target = NULL,
  citation = NULL,
  row_identifier = NULL,
  ignore_attribute = NULL,
  original_data_url = NULL,
  paper_url = NULL,
  test_server = test_server_default(),
  api_key = NULL
)
```

Arguments

`data`	(`data.frame()`) The data to upload.
`name`	(`character(1)`) The name of the dataset.
`desc`	(`character(1)`) The description of the dataset.
`license`	(`character(1)`) The license of the dataset
`default_target`	(`character(1)`) The default target variable.
`citation`	(`character(1)`) How to cite the dataset.
`row_identifier`	(`character(1)`) Whether any of the columns is a row identifier.
`ignore_attribute`	(`character(1)`) Which columns to ignore during modeling.
`original_data_url`	(character(1)) The URL of the original data set.
`paper_url`	(`character(1)`) The URL of the paper describing the data set.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.
`api_key`	(`character(1)`) The API key to perform the action, if left NULL it first tries the "mlr3oml.api_key" R option and then the environment variable `OPENMLAPIKEY`. In case `test_server` is TRUE (only relevant for developers) the test server API key is used, i.e. first the option "mlr3oml.test_api_key" and then the environment variable `TESTOPENMLAPIKEY`.

`publish_task` *Publish a task on OpenML*

Description

Publish a task on OpenML. This can also be achieved through the website.

Usage

```
publish_task(
  id,
  type,
  estimation_procedure,
  target,
  api_key = NULL,
  test_server = test_server_default()
)
```

Arguments

`id`	(`integer(1)`) The dataset id.
`type`	(`character(1)` or `integer(1)`) Can either be `"classif"` or `"regr"` or an integer indicating the task type.
`estimation_procedure`	(`integer(1)`) The id of the estimation procedure.
`target`	(`character(1)`) The target variable (if applicable).
`api_key`	(`character(1)`) The API key to perform the action, if left NULL it first tries the "mlr3oml.api_key" R option and then the environment variable `OPENMLAPIKEY`. In case `test_server` is TRUE (only relevant for developers) the test server API key is used, i.e. first the option "mlr3oml.test_api_key" and then the environment variable `TESTOPENMLAPIKEY`.
`test_server`	(`character(1)`) Whether to use the OpenML test server or public server. Defaults to value of option `"mlr3oml.test_server"`, or `FALSE` if not set.

`read_arff` *Read ARFF files*

Description

Parses a file located at `path` and returns a data.table().

Limitations:

- Only works for dense files, no support for sparse data. Use **RWeka** instead.
- Dates (even if there is no time component) are read in as POSIXct.
- The `date-format` from the ARFF specification is currently ignored. Instead, we rely on the auto-detection of **data.table**'s fread()..

Usage

```
read_arff(path)
```

Arguments

`path`	`(character(1))` Path or URI of the ARFF file, passed to `file()`.

Value

(data.table()).

`write_arff` *Write ARFF files*

Description

Writes a `data.frame()` to an ARFF file.

Limitations:

- Logicals are written as categorical features.
- POSIXct columns are converted to UTC.

Usage

```
write_arff(data, path, relation = deparse(substitute(data)))
```

Arguments

`data`	`(data.frame())` Data to write.
`path`	`(character(1))` Path or URI of the ARFF file, passed to `file()`.
`relation`	`(character(1))` Relation (name) of the data set.

Index

as_learner.OMLFlow, 4

backports::R_user_dir(), *3*
base::make.names(), *12*

data.frame(), *25*
data.table(), *26, 27*
data.table::data.table(), *12, 13*
data.table::fread(), *3*, *12*

file(), *27*
fread(), *26*

list_oml *(list_oml_collections)*, 4
list_oml_collections, 4
list_oml_data *(list_oml_collections)*, 4
list_oml_evaluations *(list_oml_collections)*, 4
list_oml_flows *(list_oml_collections)*, 4
list_oml_measures *(list_oml_collections)*, 4
list_oml_runs *(list_oml_collections)*, 4
list_oml_setups *(list_oml_collections)*, 4
list_oml_tasks *(list_oml_collections)*, 4
list_oml_tasks(), *4*

mlr3::as_benchmark_result, *2*, *10*
mlr3::as_data_backend, *2*
mlr3::as_data_backend(), *12*, *18*
mlr3::as_learner, *2*
mlr3::as_learner(), *15*
mlr3::as_learners, *10*
mlr3::as_resample_result, *2*
mlr3::as_resample_result(), *18*
mlr3::as_resampling, *2*
mlr3::as_resampling(), *18*
mlr3::as_resamplings, *10*
mlr3::as_task, *2*
mlr3::as_task(), *12*, *18*
mlr3::as_tasks, *10*
mlr3::BenchmarkResult, *10*
mlr3::DataBackend, *12*, *18*
mlr3::DataBackendDataTable, *12*
mlr3::Learner, *10*, *15*
mlr3::mlr_resamplings, *2*
mlr3::mlr_tasks, *2*
mlr3::ResampleResult, *4*, *18, 19*
mlr3::Resampling, *2*, *10*, *18*, *20*
mlr3::Task, *2*, *10*, *12*, *14*, *18*, *20, 21*
mlr3db::DataBackendDuckDB, *12*
mlr3oml *(mlr3oml-package)*, 2
mlr3oml-package, 2
mlr3oml::OMLObject, *10*, *12*, *15*, *18*, *20*

ocl, 8
ocl(), *9*
odt, 8
odt(), *11*
oflw, 9
oflw(), *15*
oml_collection, 9
oml_data, 11
oml_flow, 15
oml_object, 16
oml_run, 18
oml_run(), *18*
oml_task, 20
OMLCollection, *8*
OMLCollection *(oml_collection)*, 9
OMLData, *8*, *9*, *18*, *21*
OMLData *(oml_data)*, 11
OMLFlow, *4*, *9*, *18*
OMLFlow *(oml_flow)*, 15
OMLObject *(oml_object)*, 16
OMLRun, *22, 23*
OMLRun *(oml_run)*, 18
OMLTask, *18*, *21*, *23*
OMLTask *(oml_task)*, 20
orn, 22
otsk, 23

otsk(), *20*

POSIXct, *26*, *27*
publish_collection, 23
publish_data, 24
publish_task, 25

R6, *10*, *13*, *15*, *17*, *19*, *21*
read_arff, 26
RWeka::read.arff(), *12*

write_arff, 27