# Package 'mlr3batchmark'

*November 17, 2023*

**Title** Batch Experiments for 'mlr3'

**Version** 0.1.1

**Description** Extends the 'mlr3' package with a connector to the package
'batchtools'. This allows to run large-scale benchmark experiments on
scheduled high-performance computing clusters.

**License** LGPL-3

**URL** <https:///mlr3batchmark.mlr-org.com>,

<https://github.com/mlr-org/mlr3batchmark>

**BugReports** <https://github.com/mlr-org/mlr3batchmark/issues>

**Depends** R (>= 3.1.0), batchtools (>= 0.9.17)

**Imports** checkmate, data.table, lgr, mlr3 (>= 0.17.0), mlr3misc, uuid

**Suggests** rpart, testthat

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Marc Becker [cre, aut] (<<https://orcid.org/0000-0002-8115-0400>>),
Michel Lang [aut] (<<https://orcid.org/0000-0001-9754-0393>>)

**Maintainer** Marc Becker <marcbecker@posteo.de>

**Repository** CRAN

**Date/Publication** 2023-11-17 15:30:02 UTC

## R topics documented:

---

mlr3batchmark-package    *mlr3batchmark: Batch Experiments for 'mlr3'*

---

### Description

Extends the 'mlr3' package with a connector to the package 'batchtools'. This allows to run large-scale benchmark experiments on scheduled high-performance computing clusters.

### Author(s)

**Maintainer**: Marc Becker <marcbecker@posteo.de> (ORCID)

Authors:

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

### See Also

Useful links:

- https:///mlr3batchmark.mlr-org.com
- https://github.com/mlr-org/mlr3batchmark
- Report bugs at https://github.com/mlr-org/mlr3batchmark/issues

---

batchmark                       *Benchmark Experiments on Batch Systems*

---

### Description

This function provides the functionality to leave the interface of **mlr3** for the computation of benchmark experiments and switch over to **batchtools** for a more fine grained control over the execution.

batchmark() populates a batchtools::ExperimentRegistry with jobs in a mlr3::benchmark() fashion. Each combination of mlr3::Task and mlr3::Resampling defines a batchtools::Problem, each mlr3::Learner is an batchtools::Algorithm.

After the jobs have been submitted and are terminated, results can be collected with reduceResultsBatchmark() which returns a mlr3::BenchmarkResult and thus to return to the interface of **mlr3**.

### Usage

```
batchmark(design, store_models = FALSE, reg = batchtools::getDefaultRegistry())
```

## Arguments

design
: (`data.frame()`)
Data frame (or `data.table::data.table()`) with three columns: "task", "learner", and "resampling". Each row defines a resampling by providing a Task, Learner and an instantiated Resampling strategy. The helper function `benchmark_grid()` can assist in generating an exhaustive design (see examples) and instantiate the Resamplings per Task. Additionally, you can set the additional column 'param_values', see `benchmark_grid()`.

store_models
: (`logical(1)`)
Store the fitted model in the resulting object= Set to `TRUE` if you want to further analyse the models or want to extract information like variable importance.

reg
: batchtools::ExperimentRegistry.

## Value

`data.table()` with ids of created jobs (invisibly).

## Examples

```
tasks = list(mlr3::tsk("iris"), mlr3::tsk("sonar"))
learners = list(mlr3::lrn("classif.featureless"), mlr3::lrn("classif.rpart"))
resamplings = list(mlr3::rsmp("cv", folds = 3), mlr3::rsmp("holdout"))

design = mlr3::benchmark_grid(
  tasks = tasks,
  learners = learners,
  resamplings = resamplings
)

reg = batchtools::makeExperimentRegistry(NA)
batchmark(design, reg = reg)
batchtools::submitJobs(reg = reg)

reduceResultsBatchmark(reg = reg)
```

---

reduceResultsBatchmark
*Collect Results from batchmark*

---

## Description

Collect the results from jobs defined via `batchmark()` and combine them into a mlr3::BenchmarkResult.

Note that `ids` defaults to finished jobs (as reported by `batchtools::findDone()`). If a job threw an error, is expired or is still running, it will be ignored with this default. Just leaving these jobs out in an analysis is **not** statistically sound. Instead, try to robustify your jobs by using a fallback learner (c.f. mlr3::Learner).

**Usage**

```
reduceResultsBatchmark(
  ids = NULL,
  store_backends = TRUE,
  reg = batchtools::getDefaultRegistry()
)
```

**Arguments**

ids                    [`data.frame` or integer]
                       A `data.frame` (or `data.table`) with a column named "job.id". Alternatively,
                       you may also pass a vector of integerish job ids. If not set, defaults to the return
                       value of `findDone`. Invalid ids are ignored.

store_backends  (logical(1))
                       Keep the [DataBackend](#) of the [Task](#) in the [ResampleResult](#)? Set to `TRUE` if your
                       performance measures require a [Task](#), or to analyse results more conveniently.
                       Set to `FALSE` to reduce the file size and memory footprint after serialization. The
                       current default is `TRUE`, but this eventually will be changed in a future release.

reg                    [`Registry`]
                       Registry. If not explicitly passed, uses the default registry (see `setDefaultRegistry`).

**Value**

mlr3::BenchmarkResult.

# Index