

Package ‘mlr3batchmark’

July 23, 2025

Title Batch Experiments for 'mlr3'

Version 0.2.1

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 (>= 0.19.0), mlr3misc, uuid

Suggests renv, rpart, testthat

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Marc Becker [cre, aut] (ORCID: <<https://orcid.org/0000-0002-8115-0400>>),
Michel Lang [aut] (ORCID: <<https://orcid.org/0000-0001-9754-0393>>),
Toby Hocking [ctb] (ORCID: <<https://orcid.org/0000-0002-3146-0865>>)

Maintainer Marc Becker <marcbecker@posteo.de>

Repository CRAN

Date/Publication 2025-05-26 09:20:02 UTC

Contents

mlr3batchmark-package	2
batchmark	2
reduceResultsBatchmark	3

Index	5
--------------	----------

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](#))

Other contributors:

- Toby Hocking ([ORCID](#)) [contributor]

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(),
  renv_project = NULL
)
```

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 .
renv_project	character(1) Path to a renv project. If not NULL, the renv project is activated in the job environment.

Value

[data.table::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(),
  fun = NULL,
  unmarshal = TRUE
)
```

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).
fun	[function] Function to apply to each result. The result is passed unnamed as first argument. If NULL, the identity is used. If the function has the formal argument “job”, the Job/Experiment is also passed to the function.
unmarshal	Learner Whether to unmarshal learners that were marshaled during the execution. If TRUE all models are stored in unmarshaled form. If FALSE, all learners (that need marshaling) are stored in marshaled form.

Value

[mlr3::BenchmarkResult](#).

Index

batchmark, [2](#)
batchmark(), [3](#)
batchtools::Algorithm, [2](#)
batchtools::ExperimentRegistry, [2](#), [3](#)
batchtools::findDone(), [3](#)
batchtools::Problem, [2](#)
benchmark_grid(), [3](#)

data.frame, [4](#)
data.frame(), [3](#)
data.table, [4](#)
data.table::data.table(), [3](#)
DataBackend, [4](#)

Experiment, [4](#)

findDone, [4](#)

Job, [4](#)

Learner, [3](#), [4](#)

mlr3::benchmark(), [2](#)
mlr3::BenchmarkResult, [2–4](#)
mlr3::Learner, [2](#), [4](#)
mlr3::Resampling, [2](#)
mlr3::Task, [2](#)
mlr3batchmark (mlr3batchmark-package), [2](#)
mlr3batchmark-package, [2](#)

reduceResultsBatchmark, [3](#)
reduceResultsBatchmark(), [2](#)
Registry, [4](#)
ResampleResult, [4](#)
Resampling, [3](#)

setDefaultRegistry, [4](#)

Task, [3](#), [4](#)