Title  Analysis and Visualisation of Benchmark Experiments
Version  0.1.6
Description  Implements methods for post-hoc analysis and visualisation of benchmark experiments, for 'mlr3' and beyond.
License  LGPL-3
BugReports  https://github.com/mlr-org/mlr3benchmark/issues
Depends  R (>= 3.1.0)
Imports  checkmate, data.table, ggplot2, mlr3misc, R6
Suggests  mlr3, mlr3learners, PMCMRplus, rpart, testthat, xgboost
Encoding  UTF-8
NeedsCompilation  no
RoxygenNote  7.2.2
Author  Sonabend Raphael [aut] (<https://orcid.org/0000-0001-9225-4654>), Florian Pfisterer [aut] (<https://orcid.org/0000-0001-8867-762X>), Michel Lang [ctb] (<https://orcid.org/0000-0001-9754-0393>), Bernd Bischl [ctb] (<https://orcid.org/0000-0001-6002-6980>), Sebastian Fischer [cre, ctb] (<https://orcid.org/0000-0002-9609-3197>)
Maintainer  Sebastian Fischer <sebf.fischer@gmail.com>
Repository  CRAN
Date/Publication  2023-05-30 08:30:02 UTC

R topics documented:

mlr3benchmark-package .................................................. 2
as.BenchmarkAggr .......................................................... 2
as_benchmark_aggr ........................................................ 3
autoplot.BenchmarkAggr .................................................. 5
BenchmarkAggr .............................................................. 7
requireNamespaces ........................................................... 11
Description

Implements methods for post-hoc analysis and visualisation of benchmark experiments, for `mlr3` and beyond.

Author(s)

**Maintainer:** Sebastian Fischer <sebf.fischer@gmail.com> (ORCID) [contributor]

Authors:

- Sonabend Raphael <raphaelsonabend@gmail.com> (ORCID)
- Florian Pfisterer <pfistererf@googlemail.com> (ORCID)

Other contributors:

- Michel Lang <michellang@gmail.com> (ORCID) [contributor]
- Bernd Bischl <bernd_bischl@gmx.net> (ORCID) [contributor]

See Also

Useful links:

- [https://mlr3benchmark.mlr-org.com](https://mlr3benchmark.mlr-org.com)
- [https://github.com/mlr-org/mlr3benchmark](https://github.com/mlr-org/mlr3benchmark)

Description

This function is deprecated, use `as_benchmark_aggr()` instead.

Coercion methods to `BenchmarkAggr`. For `mlr3::BenchmarkResult` this is a simple wrapper around the `BenchmarkAggr` constructor called with `mlr3::BenchmarkResult$aggregate()`.
Usage

```r
as.BenchmarkAggr(
  obj,
  task_id = "task_id",
  learner_id = "learner_id",
  independent = TRUE,
  strip_prefix = TRUE,
  ...
)
```

Arguments

- `obj` *(mlr3::BenchmarkResult|matrix(1))*
  Passed to `BenchmarkAggr$new()`.
- `task_id`, `learner_id`, `independent`, `strip_prefix`
  See `BenchmarkAggr$initialize()`.
- `...` *ANY*
  Passed to `mlr3::BenchmarkResult$aggregate()`.

Examples

```r
df = data.frame(tasks = factor(rep(c("A", "B"), each = 5),
                            levels = c("A", "B")),
                learners = factor(paste0("L", 1:5)),
                RMSE = runif(10), MAE = runif(10))
as_benchmark_aggr(df, task_id = "tasks", learner_id = "learners")

if (requireNamespaces(c("mlr3", "rpart"))) {
  library(mlr3)
  task = tsks(c("boston_housing", "mtcars"))
  learns = lrns(c("regr.featureless", "regr.rpart"))
  bm = benchmark(benchmark_grid(task, learns, rsmp("cv", folds = 2)))

  # default measure
  as_benchmark_aggr(bm)

  # change measure
  as_benchmark_aggr(bm, measures = msr("regr.rmse"))
}
```

---

Coercions to `BenchmarkAggr`
Description

Coercion methods to `BenchmarkAggr`. For `mlr3::BenchmarkResult` this is a simple wrapper around the `BenchmarkAggr` constructor called with `mlr3::BenchmarkResult$aggregate()`.

Usage

```r
as_benchmark_aggr(
  obj,
  task_id = "task_id",
  learner_id = "learner_id",
  independent = TRUE,
  strip_prefix = TRUE,
  ...
)
```

Arguments

- **obj** (`mlr3::BenchmarkResult` | `matrix(1)`) Passed to `BenchmarkAggr$new()`.
- **task_id**, **learner_id**, **independent**, **strip_prefix** See `BenchmarkAggr$initialize()`.
- **...** ANY Passed to `mlr3::BenchmarkResult$aggregate()`.

Examples

```r
df = data.frame(tasks = factor(rep(c("A", "B"), each = 5),
                     levels = c("A", "B")),
                learners = factor(paste0("L", 1:5)),
                RMSE = runif(10), MAE = runif(10))
as_benchmark_aggr(df, task_id = "tasks", learner_id = "learners")

if (requireNamespaces(c("mlr3", "rpart"))) {
  library(mlr3)
  task = tsks(c("boston_housing", "mtcars"))
  learns = lrns(c("regr.featureless", "regr.rpart"))
  bm = benchmark(benchmark_grid(task, learns, rsmp("cv", folds = 2)))

  # default measure
  as_benchmark_aggr(bm)

  # change measure
  as_benchmark_aggr(bm, measures = msr("regr.rmse"))
}
```
Autoplot for BenchmarkAggr

Plots for BenchmarkAggr

Description

Generates plots for BenchmarkAggr, all assume that there are multiple, independent, tasks. Choices depending on the argument type:

- "mean" (default): Assumes there are at least two independent tasks. Plots the sample mean of the measure for all learners with error bars computed with the standard error of the mean.
- "box": Boxplots for each learner calculated over all tasks for a given measure.
- "fn": Plots post-hoc Friedman-Nemenyi by first calling BenchmarkAggr$friedman_posthoc and plotting significant pairs in coloured squares and leaving non-significant pairs blank, useful for simply visualising pair-wise comparisons.
- "cd": Critical difference plots (Demšar, 2006). Learners are drawn on the x-axis according to their average rank with the best performing on the left and decreasing performance going right. Any learners not connected by a horizontal bar are significantly different in performance. Critical differences are calculated as:

\[ CD = q_{\alpha} \sqrt{\left( \frac{k(k+1)}{6N} \right)} \]

Where \( q_{\alpha} \) is based on the studentized range statistic. See references for further details. It’s recommended to crop white space using external tools, or function image_trim() from package magick.

Usage

```r
## S3 method for class 'BenchmarkAggr'
autoplot(
  object,
  type = c("mean", "box", "fn", "cd"),
  meas = NULL,
  level = 0.95,
  p.value = 0.05,
  minimize = TRUE,
  test = "nem",
  baseline = NULL,
  style = 1L,
  ratio = 1/7,
  col = "red",
  friedman_global = TRUE,
  ...
)
```

Arguments

- **object** *(BenchmarkAggr)*
  - The benchmark aggregation object.

- **type** *(character(1))*
  - Type of plot, see description.

- **meas** *(character(1))*
  - Measure to plot, should be in obj$measures, can be NULL if only one measure is in obj.

- **level** *(numeric(1))*
  - Confidence level for error bars for type = "mean"

- **p.value** *(numeric(1))*
  - What value should be considered significant for type = "cd" and type = "fn".

- **minimize** *(logical(1))*
  - For type = "cd", indicates if the measure is optimally minimized. Default is TRUE.

- **test** *(character(1))*
  - For type = "cd", critical differences are either computed between all learners (test = "nemenyi"), or to a baseline (test = "bd"). Bonferroni-Dunn usually yields higher power than Nemenyi as it only compares algorithms to one baseline. Default is "nemenyi".

- **baseline** *(character(1))*
  - For type = "cd" and test = "bd" a baseline learner to compare the other learners to, should be in $learners, if NULL then differences are compared to the best performing learner.

- **style** *(integer(1))*
  - For type = "cd" two ggplot styles are shipped with the package (style = 1 or style = 2), otherwise the data can be accessed via the returned ggplot.

- **ratio** *(numeric(1))*
  - For type = "cd" and style = 1, passed to ggplot2::coord_fixed(), useful for quickly specifying the aspect ratio of the plot, best used with ggsave().

- **col** *(character(1))*
  - For type = "fn", specifies color to fill significant tiles, default is "red".

- **friedman_global** *(logical(1))*
  - Should a friedman global test be performed for type = "cd" and type = "fn"? If FALSE, a warning is issued in case the corresponding friedman posthoc test fails instead of an error. Default is TRUE (raises an error if global test fails).

- **...** *(ANY)*
  - Additional arguments, currently unused.

Value

The generated plot.
References


Examples

```r
if (requireNamespaces(c("mlr3learners", "mlr3", "rpart", "xgboost"))) {
  library(mlr3)
  library(mlr3learners)
  library(ggplot2)
  set.seed(1)
  task = tsks(c("iris", "sonar", "wine", "zoo"))
  learns = lrns(c("classif.featureless", "classif.rpart", "classif.xgboost"))
  bm = benchmark(benchmark_grid(task, learns, rsmp("cv", folds = 3)))
  obj = as_benchmark_aggr(bm)

  # mean and error bars
  autoplot(obj, type = "mean", level = 0.95)

  if (requireNamespace("PMCMRplus", quietly = TRUE)) {
    # critical differences
    autoplot(obj, type = "cd", style = 1)
    autoplot(obj, type = "cd", style = 2)

    # post-hoc friedman-nemenyi
    autoplot(obj, type = "fn")
  }
}
```

BenchmarkAggr

**Aggregated Benchmark Result Object**

**Description**

An R6 class for aggregated benchmark results.

**Details**

This class is used to easily carry out and guide analysis of models after aggregating the results after resampling. This can either be constructed using `mlr3` objects, for example the result of `mlr3::BenchmarkResult$aggregate` or via `as_benchmark_aggr`, or by passing in a custom dataset of results. Custom datasets must include at the very least, a character column for learner ids, a character column for task ids, and numeric columns for one or more measures. Currently supported for multiple independent datasets only.
Active bindings

- **data** (data.table::data.table)
  - Aggregated data.
- **learners** (character())
  - Unique learner names.
- **tasks** (character())
  - Unique task names.
- **measures** (character())
  - Unique measure names.
- **nlrns** (integer())
  - Number of learners.
- **ntasks** (integer())
  - Number of tasks.
- **nmeas** (integer())
  - Number of measures.
- **nrow** (integer())
  - Number of rows.
- **col_roles** (character())
  - Column roles, currently cannot be changed after construction.

Methods

**Public methods:**

- `BenchmarkAggr$new()`
- `BenchmarkAggr$print()`
- `BenchmarkAggr$summary()`
- `BenchmarkAggr$rank_data()`
- `BenchmarkAggr$friedman_test()`
- `BenchmarkAggr$friedman_posthoc()`
- `BenchmarkAggr$subset()`
- `BenchmarkAggr$clone()`

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

*Usage:*

```r
BenchmarkAggr$new(
  dt,
  task_id = "task_id",
  learner_id = "learner_id",
  independent = TRUE,
  strip_prefix = TRUE,
  ...
)
```

*Arguments:*
dt (matrix(1))
  'matrix like object coercable to \texttt{data.table::data.table}, should include column names "task_id" and "learner_id", and at least one measure (numeric). If ids are not already factors then co-
erced internally.

task_id (character(1))
  String specifying name of task id column.

learner_id (character(1))
  String specifying name of learner id column.

independent (logical(1))
  Are tasks independent of one another? Affects which tests can be used for analysis.

strip_prefix (logical(1))
  If TRUE (default) then mlr prefixes, e.g. \texttt{regr.}, \texttt{classif.}, are automatically stripped from
  the learner_id.

... ANY
  Additional arguments, currently unused.

\textbf{Method} \texttt{print()}: Prints the internal data via \texttt{data.table::print.data.table}.

\textit{Usage}:
BenchmarkAggr$\texttt{print}(...)

\textit{Arguments}:
... ANY
  Passed to \texttt{data.table::print.data.table}.

\textbf{Method} \texttt{summary()}: Prints the internal data via \texttt{data.table::print.data.table}.

\textit{Usage}:
BenchmarkAggr$\texttt{summary}(...)

\textit{Arguments}:
... ANY
  Passed to \texttt{data.table::print.data.table}.

\textbf{Method} \texttt{rank_data()}: Ranks the aggregated data given some measure.

\textit{Usage}:
BenchmarkAggr$\texttt{rank_data}(meas = NULL, minimize = TRUE, task = NULL, ...)

\textit{Arguments}:
meas (character(1))
  Measure to rank the data against, should be in \$\texttt{measures}. Can be NULL if only one measure
  in data.

minimize (logical(1))
  Should the measure be minimized? Default is TRUE.

task (character(1))
  If NULL then returns a matrix of ranks where columns are tasks and rows are learners, oth-
erwise returns a one-column matrix of a specified task, should be in \$\texttt{tasks}.

... ANY ANY
  Passed to \texttt{data.table::frank()}.
**Method** friedman_test(): Computes Friedman test over all tasks, assumes datasets are independent.

*Usage:*

```r
BenchmarkAggr$friedman_test(meas = NULL, p.adjust.method = NULL)
```

*Arguments:*

- `meas` (character(1))
  - Measure to rank the data against, should be in `$measures`. If no measure is provided then returns a matrix of tests for all measures.
- `p.adjust.method` (character(1))
  - Passed to `p.adjust` if `meas = NULL` for multiple testing correction. If `NULL` then no correction applied.

**Method** friedman_posthoc(): Posthoc Friedman Nemenyi tests. Computed with PMCMRplus::frdAllPairsNemenyiTest. If global `$friedman_test` is non-significant then this is returned and no post-hocs computed. Also returns critical difference

*Usage:*

```r
BenchmarkAggr$friedman_posthoc(  
  meas = NULL,  
  p.value = 0.05,  
  friedman_global = TRUE  
)
```

*Arguments:*

- `meas` (character(1))
  - Measure to rank the data against, should be in `$measures`. Can be `NULL` if only one measure in data.
- `p.value` (numeric(1))
  - `p.value` for which the global test will be considered significant.
- `friedman_global` (logical(1))
  - Should a friedman global test be performed before conducting the posthoc test? If `FALSE`, a warning is issued in case the corresponding friedman global test fails instead of an error. Default is `TRUE` (raises an error if global test fails).

**Method** subset(): Subsets the data by given tasks or learners. Returns data as `data.table::data.table`.

*Usage:*

```r
BenchmarkAggr$subset(task = NULL, learner = NULL)
```

*Arguments:*

- `task` (character())
  - Task(s) to subset the data by.
- `learner` (character())
  - Learner(s) to subset the data by.

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

*Usage:*

```r
BenchmarkAggr$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.
References

```r
format_bib("demsar_2006")
```

Examples

```r
# Not restricted to mlr3 objects
df = data.frame(tasks = factor(rep(c("A", "B"), each = 5),
  levels = c("A", "B"),
  learners = factor(paste0("L", 1:5)),
  RMSE = runif(10), MAE = runif(10))
as_benchmark_aggr(df, task_id = "tasks", learner_id = "learners")
```

```r
if (requireNamespaces(c("mlr3", "rpart"))) {
  library(mlr3)
  task = tsks(c("boston_housing", "mtcars"))
  learns = lrns(c("regr.featureless", "regr.rpart"))
  bm = benchmark(benchmark_grid(task, learns, rsmp("cv", folds = 2)))

  # coercion
  as_benchmark_aggr(bm)
}
```

---

**requireNamespaces**  
**Helper Vectorizing requireNamespace**

**Description**

Internal helper function for documentation.

**Usage**

```r
requireNamespaces(x)
```

**Arguments**

- `x` Packages to check.

**Value**

A logical(1), indicating whether all required packages are available.
Index

as.BenchmarkAggr, 2
as_benchmark_aggr, 3, 7
as_benchmark_aggr(), 2
autoplot.BenchmarkAggr, 5

BenchmarkAggr, 2–6, 7

data.table::data.table, 8–10
data.table::frank(), 9
data.table::print.data.table, 9

ggplot2::coord_fixed(), 6
ggsave(), 6

mlr3::BenchmarkResult, 2–4, 7
mlr3benchmark (mlr3benchmark-package), 2
mlr3benchmark-package, 2

p.adjust, 10
PMCMRplus::frdAllPairsNemenyiTest, 10

R6, 8
requireNamespaces, 11