Package: BenchHub (via r-universe)

November 25, 2026

Title Comprehensive Collection of Curated Benchmarking Datasets and their Evaluation

Version 1.0.0

Description The trio is the combination of a data set, a metric and supporting evidence which provides some best case scenario, if not the ground truth itself. BenchHub has data downloaders for FigShare, G.E.O., and ExperimentHub. Caching is used to avoid lengthy downloads after the first time a data set is accessed. The user may also specify their own data set and supporting evidence. The Benchmark Insights module provides functionality for comparing and contrasting the performance of alternative algorithms.

biocViews Infrastructure, DataImport, DataRepresentation, Software, WorkflowStep, Visualization

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.5.0)

Imports cli, fs, glue, R6, rlang, httr2, stringr, purrr, dplyr, reshape2, utils, curl, googlesheets4, survAUC, Hmisc, ggrepel, ggsci, ggcorrplot, broom, dotwhisker, splitTools, magrittr, withr, ggplot2, data.table, jsonlite

Suggests anndata, readr, GEOquery, ExperimentHub, zen4R, knitr, ks, rmarkdown, SingleCellExperiment, SummarizedExperiment, survival, cvTools, funkyheatmap, testthat (>= 3.0.0), clusterProfiler, DOSE, DO.db, EnsDb.Hsapiens.v86, BiocStyle, tidyverse, glmnet, scran, scuttle, edgeR

VignetteBuilder knitr

Config/testthat/edition 3

BugReports https://github.com/SydneyBioX/BenchHub/issues

URL https://sydneybiox.github.io/BenchHub/

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:19 UTC

RemoteUrl https://github.com/bioc/BenchHub

RemoteRef RELEASE_3_24

RemoteSha fdc9c929c57556db8ea845eac846b4e40f086e59

Contents

ARImetric . . . 3
balAccMetric . . . 4
balErrMetric . . . 4
beggCIndexMetric . . . 5
BenchmarkInsights . . . 6
BenchmarkStudy . . . 8
brierScoreMetric . . . 12
buildDatasetEvidenceSubmission . . . 12
buildDatasetSubmission . . . 13
buildDatasetTaskMetricSubmission . . . 15
buildDatasetTaskSubmission . . . 16
buildMetricSubmission . . . 17
buildStudySubmission . . . 18
buildStudySubmissionPayload . . . 19
buildTrioSubmission . . . 20
buildTrioSubmissionPayload . . . 21
collectDatasetSubmissionInfo . . . 22
collectDatasetTaskMetricSubmission . . . 23
collectEvidenceSubmissionInfo . . . 24
collectMetricSubmissionInfo . . . 25
collectStudySubmissionInfo . . . 26
collectTaskSubmissionInfo . . . 28
downloadSubmissionStudy . . . 29
downloadSubmissionTrio . . . 30
getSubmissionStudy . . . 30
getSubmissionStudyDatasets . . . 31
ghCIndexMetric . . . 32
harrelCIndexMetric . . . 33
interactivePrepareStudySubmission . . . 33
interactivePrepareStudyUpdateSubmission . . . 34
JSDmetric . . . 35
kdeMetric . . . 36
listCuratedTrioDatasets . . . 36
listCuratedTrioStudies . . . 37
listSubmissionStudies . . . 37
listSubmissionStudyDatasets . . . 38

lubomski_microbiome_data 38
macroF1Metric 39
macroPrecMetric 39
macroRecMetric 40
MCCmetric 40
microF1Metric 41
microPrecMetric 42
microRecMetric 42
MSEmetric 43
NMImetric 43
prepareStudySubmission 44
prepareStudyUpdateSubmission 46
prepareTrioSubmissionBundle 48
prepareTrioSubmissionFiles 49
prepareTrioSubmissionMetrics 51
RMSEmetric 52
studySubmissionToJSON 53
submitStudySubmission 54
submitTrioSubmission 55
timeDependentAUCMetric 56
Trio 57
trioSubmissionToJSON 61
unoCIndexMetric 62
writeSubmission 63

Index **66**

`ARImetric` *Adjusted Rand Index (ARI) Metric*

Description

Computes the adjusted Rand index between two cluster labelings.

Usage

```
ARImetric(evidence, predicted)
```

Arguments

`evidence`	The true labels.
`predicted`	The predicted labels.

Value

The adjusted Rand index.

Examples

```
evidence <- factor(c("A", "A", "B", "B"))
predicted <- factor(c("A", "A", "B", "B"))
ARImetric(evidence, predicted)
```

`balAccMetric` *Balanced Accuracy Metric*

Description

Computes the balanced accuracy of the predictions.

Usage

```
balAccMetric(evidence, predicted)
```

Arguments

`evidence`	The true labels.
`predicted`	The predicted labels.

Value

The balanced accuracy.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
balAccMetric(evidence, predicted)
```

`balErrMetric` *Balanced Error Metric*

Description

Computes the balanced error of the predictions.

Usage

```
balErrMetric(evidence, predicted)
```

Arguments

`evidence`	The true labels.
`predicted`	The predicted labels.

Value

The balanced error.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
balErrMetric(evidence, predicted)
```

`beggCIndexMetric` *Begg's C-Index Metric*

Description

Computes Begg's C-Index for survival analysis.

Usage

```
beggCIndexMetric(evidence, predicted)
```

Arguments

`evidence`	The true survival times and event indicators.
`predicted`	The predicted survival times.

Value

Begg's C-Index.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
  event = c(1, 0, 1, 0))  # Testing
)
# Predicted risk scores
predicted <- list(
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
beggCIndexMetric(evidence, predicted)
```

`BenchmarkInsights` *A BenchmarkInsights object*

Description

An object containing a benchmark result for evaluating analytical tasks.

Value

A BenchmarkInsights object.

Public fields

`evalSummary` The evaluation summary is stored by dataframe, where each row is the methodd identifier, each column is the metric used in the evaluation task and related information.

`metadata` A dataframe to store metadata for the benchmark.

Methods

Public methods:

- `BenchmarkInsights$new()`
- `BenchmarkInsights$addevalSummary()`
- `BenchmarkInsights$addMetadata()`
- `BenchmarkInsights$getHeatmap()`
- `BenchmarkInsights$getLineplot()`
- `BenchmarkInsights$getScatterplot()`
- `BenchmarkInsights$getBoxplot()`
- `BenchmarkInsights$getCorplot()`
- `BenchmarkInsights$getForestplot()`
- `BenchmarkInsights$clone()`

Method `new()`**:** Create a BenchmarkInsights object

Usage:

```
BenchmarkInsights$new(evalResult = NULL)
```

Arguments:

`evalResult` A dataframe containing initial evaluation results with columns such as datasetID, evidence, metric, and result.

Method `addevalSummary()`**:** Add additional evaluation summary to the existing evalSummary

Usage:

```
BenchmarkInsights$addevalSummary(additional_evalResult)
```

Arguments:

`additional_evalResult` A dataframe containing additional evaluation results to be appended.

Method `addMetadata()`**:** Add metadata to the BenchmarkInsights object

Usage:

```
BenchmarkInsights$addMetadata(metadata)
```

Arguments:

`metadata` A dataframe containing metadata information.

Method `getHeatmap()`**:** Creates a heatmap from the evaluation summary by averaging results across datasets.

Usage:

```
BenchmarkInsights$getHeatmap()
```

Returns: A heatmap object.

Method `getLineplot()`**:** Creates a line plot for the given x and y variables, with an optional grouping and fixed x order.

Usage:

```
BenchmarkInsights$getLineplot(order = NULL, metricVariable)
```

Arguments:

`order` An optional vector specifying the order of x-axis values.

`metricVariable` Specify subset value in metric column.

Returns: A ggplot2 line plot object.

Method `getScatterplot()`**:** Creates a scatter plot for the same evidence, with an two method metrics.

Usage:

```
BenchmarkInsights$getScatterplot(variables)
```

Arguments:

`variables` A character vector of length two specifying the metric names to be used for the x and y axes.

Returns: A ggplot2 line plot object.

Method `getBoxplot()`**:** Creates boxplot plots for the mutiple evidence, different method, one metric.

Usage:

```
BenchmarkInsights$getBoxplot(metricVariable, evidenceVariable)
```

Arguments:

`metricVariable` Specify subset value in metric column.

`evidenceVariable` Specify subset value in evidence column.

Returns: A ggplot2 line plot object.

Method `getCorplot()`**:** Creates a correlation plot based on the provided evaluation summary and the specified input type (either "evidence", "metric", or "method"). The correlation plot shows the pairwise correlation between results for different categories (evidence, metric, or method).

Usage:

```
BenchmarkInsights$getCorplot(input_type)
```

Arguments:

`input_type` A string that specifies the input type for generating the correlation plot. It must be either "evidence", "metric", or "method".

Returns: A ggplot2 correlation plot object. The plot visualizes the correlation matrix using ggcorrplot with aesthetic enhancements like labeled values and angled axis text.

Method `getForestplot()`**:** This function generates a forest plot using linear models based on the comparison between groups in the provided evaluation summary. The plot is created using dotwhisker and broom packages, with custom grouping and labeling.

Usage:

```
BenchmarkInsights$getForestplot(input_group, input_model)
```

Arguments:

`input_group` A string specifying the grouping variable (only "datasetID", "method", or "evidence" allowed).

`input_model` A string specifying the model variable (only "datasetID", "method", or "evidence" allowed).

Returns: A forest plot showing the comparison of models across groups.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
BenchmarkInsights$new()
```

`BenchmarkStudy` *BenchmarkStudy Class*

Description

This class manages a collection of benchmark trios and mapping functions. It allows adding new trios, mapping functions, and running mappings on data.

Value

A `BenchmarkStudy` object.

Public fields

`name` A character string to name the study.

`trios` A list to store benchmark trios.

`mappingFunctions` A list to store mapping functions with metadata.

`description` A character string describing the study.

`version` Integer specifying the version of the study.

Methods

Public methods:

- `BenchmarkStudy$new()`
- `BenchmarkStudy$addTrio()`
- `BenchmarkStudy$addMappingFunction()`
- `BenchmarkStudy$runMapping()`
- `BenchmarkStudy$getMappingFunctionDocumentation()`
- `BenchmarkStudy$printMappingFunctionDocumentation()`
- `BenchmarkStudy$listMappingFunctions()`
- `BenchmarkStudy$generateVignetteTemplate()`
- `BenchmarkStudy$evaluate()`
- `BenchmarkStudy$writeBenchmarkStudy()`
- `BenchmarkStudy$print()`
- `BenchmarkStudy$clone()`

Method `new()`:

Usage:

```
BenchmarkStudy$new(
  name = NULL,
  trios = list(),
  fetchFromCtd = FALSE,
  version = NULL
)
```

Arguments:

`name` A character string to name the study. If fetchFromCtd is TRUE, this name will be used to fetch the study from the submission database.

`trios` A list of Trio objects to initialize the study.

`fetchFromCtd` Logical indicating whether to fetch study details from the submission database.

`version` Optional integer specifying which version of the study to fetch (when fetchFromCtd is TRUE).

Method `addTrio()`: Add a new trio to the study

Usage:

```
BenchmarkStudy$addTrio(trioObject)
```

Arguments:

`trioObject` A Trio object to be added.

Method `addMappingFunction()`**:** Add a mapping function with metadata

Usage:

```
BenchmarkStudy$addMappingFunction(
  name,
  func,
  inputDescription,
  outputDescription,
  exampleUsage = NULL
)
```

Arguments:

`name` A character string to name the mapping function.

`func` A function that takes data as input and returns transformed data.

`inputDescription` A character string describing the input data.

`outputDescription` A character string describing the output data.

`exampleUsage` An optional character string showing example usage of the function.

Method `runMapping()`**:** Apply a mapping function to data

Usage:

```
BenchmarkStudy$runMapping(mappingName, data)
```

Arguments:

`mappingName` A character string naming the mapping function to apply.

`data` The data to which the mapping function will be applied.

Returns: The transformed data after applying the mapping function.

Method `getMappingFunctionDocumentation()`**:** Documentation getter for mapping function

Usage:

```
BenchmarkStudy$getMappingFunctionDocumentation(mappingName)
```

Arguments:

`mappingName` A character string naming the mapping function.

Returns: A list containing the input description, output description, and example usage.

Method `printMappingFunctionDocumentation()`**:** Print the documentation for a mapping function in a human-readable format.

Usage:

```
BenchmarkStudy$printMappingFunctionDocumentation(mappingName)
```

Arguments:

`mappingName` A character string naming the mapping function.

Returns: Prints inputDescription, outputDescription, exampleUsage

Method `listMappingFunctions()`**:** available options

Usage:

```
BenchmarkStudy$listMappingFunctions()
```

Returns: A character vector of mapping function names.

Method `generateVignetteTemplate()`**:** Generate R Markdown vignette template

Usage:

```
BenchmarkStudy$generateVignetteTemplate(
  outputPath = "benchmark_study_template.Rmd"
)
```

Arguments:

`outputPath` A character string specifying the path to save the vignette template.

Method `evaluate()`**:** Evaluate a trio with input data

Usage:

```
BenchmarkStudy$evaluate(trioName, input)
```

Arguments:

`trioName` A character string naming the trio to evaluate.

`input` The input data to evaluate the trio against.

Returns: The evaluation result from the trio.

Method `writeBenchmarkStudy()`**:** Write the BenchmarkStudy metadata to Curated Trio Datasets sheet.

Usage:

```
BenchmarkStudy$writeBenchmarkStudy()
```

Method `print()`**:** Print method to display key information about the BenchmarkStudy object.

Usage:

```
BenchmarkStudy$print()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
study$name
```

`brierScoreMetric` *Brier Score Metric*

Description

Computes the Brier score for survival analysis.

Usage

```
brierScoreMetric(evidence, predicted)
```

Arguments

`evidence`	The true survival times and event indicators.
`predicted`	The predicted survival times.

Value

The Brier score.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
  event = c(1, 0, 1, 0))  # Testing
)
# Predicted risk scores
predicted <- list(
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
brierScoreMetric(evidence, predicted)
```

`buildDatasetEvidenceSubmission`
Build DatasetEvidence submission rows from a Trio object

Description

This helper creates one `DatasetEvidence` row per supporting evidence entry in the Trio object. It does not upload anything and does not modify the existing writeCTD() path.

Usage

```
buildDatasetEvidenceSubmission(
  trio,
  datasetTaskID,
  evidenceName = names(trio$evidence),
  evidenceType = NA_character_
)
```

Arguments

trio	A Trio object.
datasetTaskID	Task identifier to link evidence rows to. Defaults to generated task IDs when not supplied.
evidenceName	Character vector of Trio evidence names to include. Defaults to all evidence in trio.
evidenceType	Either "columns" or "figshare" indicating where to find the evidence.

Value

A data.frame matching the proposed DatasetEvidence table schema. evidenceID is left as NA.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
buildDatasetEvidenceSubmission(
  trio,
  datasetTaskID = "task_001",
  evidenceType = "experimental_ground_truth"
)
```

---

buildDatasetSubmission

Build a Dataset submission row from a Trio object

Description

This helper is the first step in the new submission workflow. It does not upload anything and does not modify the existing writeCTD() path.

Usage

```
buildDatasetSubmission(
  trio,
  name = private_submission_dataset_name(trio),
  dataType,
  dataModality,
  technology,
  description = trio$description,
  doi = NA_character_,
  organism = NA_character_,
  tissue,
  status
)
```

Arguments

`trio`	A `Trio` object.
`name`	Dataset name for submission. Defaults to the existing Trio name, falling back to `trio$dataSourceID` when needed.
`dataType`	One of `"omics"` or `"clinical"`.
`dataModality`	One of `"genomics"`, `"transcriptomics"`, `"epigenomics"`, `"proteomics"`, `"metabolomics"`, `"patient"`, or `"other"`.
`technology`	Free-text technology description.
`description`	Dataset description. Defaults to `trio$description` if present.
`doi`	DOI reference for the dataset.
`organism`	Free-text organism label.
`tissue`	Free-text tissue label.
`status`	One of `"healthy"`, `"diseased"`, `"developmental"`, or `"other"`.

Value

A one-row `data.frame` matching the proposed `Dataset` table schema. `datasetID` is left as `NA` because it is expected to be generated by the submission backend.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
```

```
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
buildDatasetSubmission(
  trio,
  dataType = "omics",
  dataModality = "transcriptomics",
  technology = "RNA-seq",
  tissue = "blood",
  status = "healthy"
)
```

`buildDatasetTaskMetricSubmission` *Build DatasetTaskMetric submission rows from a Trio object*

Description

This helper creates one `DatasetTaskMetric` row for each task-metric pairing. It does not upload anything and does not modify the existing writeCTD() path.

Usage

```
buildDatasetTaskMetricSubmission(
  trio,
  datasetTaskID,
  taskMetrics = NULL,
  evidenceTaskID = NULL,
  evidenceName = names(trio$evidence)
)
```

Arguments

`trio`	A `Trio` object.
`datasetTaskID`	Task identifier(s) to link metrics to. Defaults to `NA` generated task IDs when not supplied.
`taskMetrics`	A named list mapping each `datasetTaskID` to its metric IDs. If omitted, metrics are inferred from the evidence assigned to each task via `evidenceTaskID`.
`evidenceTaskID`	Character vector mapping each Trio evidence entry to a `datasetTaskID`. Used when `taskMetrics` is not supplied.
`evidenceName`	Character vector of Trio evidence names corresponding to `evidenceTaskID`. Defaults to all evidence in `trio`.

Value

A `data.frame` matching the proposed `DatasetTaskMetric` table schema. `datasetTaskMetricID` and `metricID` are left as `NA`.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
buildDatasetTaskMetricSubmission(
  trio,
  datasetTaskID = "task_001",
  evidenceTaskID = "task_001"
)
```

---

buildDatasetTaskSubmission

Build DatasetTask submission rows from a Trio object

Description

This helper creates one `DatasetTask` row per supporting evidence name in the Trio object. It does not upload anything and does not modify the existing writeCTD() path.

Usage

```
buildDatasetTaskSubmission(
  trio,
  datasetID = NA_character_,
  taskStage,
  taskType,
  taskName
)
```

Arguments

`trio`	A `Trio` object.
`datasetID`	Dataset identifier to link tasks to. Defaults to `NA` so the backend can fill it later.

taskStage	A character vector giving the task stage for each task row.
taskType	A character vector giving the task type for each task row.
taskName	A character vector giving the task name for each task row.

Value

A data.frame matching the proposed DatasetTask table schema. datasetTaskID is left as NA.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
buildDatasetTaskSubmission(
  trio,
  taskStage = "prediction",
  taskType = "classification",
  taskName = "class_prediction"
)
```

---

buildMetricSubmission *Build Metric submission rows from a Trio object*

Description

This helper creates one Metric row per metric in the Trio object. It does not upload anything and does not modify the existing writeCTD() path.

Usage

```
buildMetricSubmission(trio)
```

Arguments

trio	A Trio object.

Value

A data.frame matching the proposed Metric table schema.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
buildMetricSubmission(trio)
```

`buildStudySubmission` *Build Study and StudyDataset submission tables*

Description

Build Study and StudyDataset submission tables

Usage

```
buildStudySubmission(
  study,
  datasetIDs,
  existing_studies = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  version = NULL,
  type = NULL,
  protocolGist = "",
  mappingFunctions = ""
)
```

Arguments

`study`	A `BenchmarkStudy` object.
`datasetIDs`	Character vector of existing dataset IDs to link.
`existing_studies`	Optional data frame of current Study rows.
`ss`	Submission spreadsheet ID or URL. Used when `existing_studies` is not supplied.
`version`	Optional version override.
`type`	Optional type override. Must be `"original"` or `"update"` when provided.

protocolGist Optional protocol gist URL.

mappingFunctions
Optional mapping functions gist URL.

Value

A named list containing Study and StudyDataset.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
buildStudySubmission(
  study,
  datasetIDs = "dataset_001",
  existing_studies = existing_studies
)
```

---

buildStudySubmissionPayload

Convert a Study submission to payload structure

Description

Convert a Study submission to payload structure

Usage

```
buildStudySubmissionPayload(submission)
```

Arguments

submission A submission object returned by buildStudySubmission().

Value

A named list with a top-level payload entry.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
submission <- buildStudySubmission(
  study,
  datasetIDs = "dataset_001",
  existing_studies = existing_studies
)
payload <- buildStudySubmissionPayload(submission)
names(payload)
```

---

buildTrioSubmission *Build a combined Trio submission object*

Description

This helper assembles the five Trio-focused submission tables into a single list so it can be inspected locally before any upload step is added.

Usage

```
buildTrioSubmission(trio, dataset_args, task_args, evidence_task_map)
```

Arguments

trio	A Trio object.
dataset_args	Named list of arguments passed to buildDatasetSubmission(), excluding trio.
task_args	Named list of arguments passed to buildDatasetTaskSubmission(), excluding trio.
evidence_task_map	Character vector mapping Trio evidence names to the corresponding task names in the submission. Names must be evidence names from trio$evidence, values must be task names supplied in task_args.

Value

A named list containing Dataset, DatasetTask, DatasetEvidence, Metric, DatasetTaskMetric, and submission_links.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
dataset_args <- list(
  dataType = "omics",
  dataModality = "transcriptomics",
  technology = "RNA-seq",
  tissue = "blood",
  status = "healthy"
)
task_args <- list(
  taskStage = "prediction",
  taskType = "classification",
  taskName = "class_prediction"
)
submission <- buildTrioSubmission(
  trio = trio,
  dataset_args = dataset_args,
  task_args = task_args,
  evidence_task_map = c(class_labels = "class_prediction")
)
names(submission)
```

---

buildTrioSubmissionPayload

Convert a combined Trio submission to payload structure

Description

This helper converts the output of `buildTrioSubmission()` into the nested list structure used by the JSON payload in the prototype submission script.

Usage

```
buildTrioSubmissionPayload(submission)
```

Arguments

`submission`	A submission object returned by `buildTrioSubmission()`.

Value

A named list with a top-level payload entry ready for jsonlite::toJSON(..., auto_unbox = TRUE).

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
submission <- buildTrioSubmission(
  trio = trio,
  dataset_args = list(
    dataType = "omics",
    dataModality = "transcriptomics",
    technology = "RNA-seq",
    tissue = "blood",
    status = "healthy"
  ),
  task_args = list(
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  ),
  evidence_task_map = c(class_labels = "class_prediction")
)
payload <- buildTrioSubmissionPayload(submission)
names(payload)
```

---

collectDatasetSubmissionInfo

Collect Dataset submission metadata interactively

Description

This helper gathers the dataset-level fields needed for the Dataset table. It reuses Trio metadata when already available and only prompts for the remaining values in interactive sessions.

Usage

```
collectDatasetSubmissionInfo(trio, defaults = list())
```

Arguments

trio	A Trio object.
defaults	Optional named list of pre-filled values. Any non-NULL value here bypasses the interactive prompt for that field.

Value

A named list ready to pass as dataset_args to buildDatasetSubmission().

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
collectDatasetSubmissionInfo(
  trio,
  defaults = list(
    dataType = "omics",
    dataModality = "transcriptomics",
    technology = "RNA-seq",
    tissue = "blood",
    status = "healthy"
  )
)
```

---

collectDatasetTaskMetricSubmission

Collect DatasetTaskMetric submission rows from Trio assignments

---

Description

This helper builds the DatasetTaskMetric linking table automatically from the evidence-to-task assignments and the metric links already stored inside the Trio object.

Usage

```
collectDatasetTaskMetricSubmission(trio, evidence_args, task_args)
```

Arguments

trio	A Trio object.
evidence_args	Named list returned by collectEvidenceSubmissionInfo().
task_args	Named list returned by collectTaskSubmissionInfo().

Value

A data.frame matching the DatasetTaskMetric schema.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
collectDatasetTaskMetricSubmission(
  trio,
  evidence_args = list(
    datasetTaskID = "task_001",
    evidenceName = "class_labels"
  ),
  task_args = list(
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  )
)
```

---

collectEvidenceSubmissionInfo

Collect DatasetEvidence submission metadata interactively

Description

This helper gathers the evidence-to-task assignment and evidence type for each supporting evidence item in a Trio.

Usage

```
collectEvidenceSubmissionInfo(trio, task_args, defaults = list())
```

Arguments

trio	A Trio object.
task_args	Named list returned by collectTaskSubmissionInfo().
defaults	Optional named list with entries taskName and evidenceType. Each entry may be length 1 or length equal to the number of supporting evidence items in trio.

Value

A named list with datasetTaskID, evidenceName, evidenceType, and evidence_task_map.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
task_args <- list(
  taskStage = "prediction",
  taskType = "classification",
  taskName = "class_prediction"
)
collectEvidenceSubmissionInfo(
  trio,
  task_args = task_args,
  defaults = list(
    taskName = "class_prediction",
    evidenceType = "experimental_ground_truth"
  )
)
```

---

collectMetricSubmissionInfo

Collect Metric submission metadata interactively

Description

This helper gathers user-supplied metric classification for each metric in a Trio. The metric names and wrappers still come from the Trio object and the package helper logic.

Usage

```
collectMetricSubmissionInfo(trio, defaults = list())
```

Arguments

trio	A Trio object.
defaults	Optional named list with entry metricType. It may be length 1 or length equal to the number of metrics in the Trio.

Value

A named list ready to merge into the Metric submission table.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
collectMetricSubmissionInfo(
  trio,
  defaults = list(metricType = "label_based")
)
```

---

collectStudySubmissionInfo

Collect Study submission metadata

Description

Collect Study submission metadata

Usage

```
collectStudySubmissionInfo(
  study,
  datasetIDs = NULL,
  available_datasets = NULL,
  existing_studies = NULL,
```

```
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  defaults = list()
)
```

Arguments

study	A BenchmarkStudy object.
datasetIDs	Optional character vector of dataset IDs. When NULL, datasets can be inferred from study$trios or selected interactively.
available_datasets	Optional data frame of available Dataset rows.
existing_studies	Optional data frame of current Study rows.
ss	Submission spreadsheet ID or URL.
defaults	Optional named list with entries such as datasetIDs, version, type, protocolGist, and mappingFunctions.

Value

A named list ready to pass into buildStudySubmission().

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
available_datasets <- data.frame(
  datasetID = "dataset_001",
  name = "example_dataset",
  stringsAsFactors = FALSE
)
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
collectStudySubmissionInfo(
  study,
  datasetIDs = "dataset_001",
  available_datasets = available_datasets,
  existing_studies = existing_studies
)
```

collectTaskSubmissionInfo *Collect DatasetTask submission metadata interactively*

Description

This helper gathers the task-level fields needed for the DatasetTask table. It does not modify the Trio object.

Usage

```
collectTaskSubmissionInfo(trio, defaults = list(), n_tasks = NULL)
```

Arguments

trio	A Trio object.
defaults	Optional named list of pre-filled task values. Supported entries are taskStage, taskType, and taskName. Each entry may be length 1 or length n_tasks.
n_tasks	Number of tasks to define. If NULL, interactive sessions prompt for this value, while non-interactive sessions infer it from the longest defaults vector.

Value

A named list ready to pass as task_args to buildDatasetTaskSubmission().

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
collectTaskSubmissionInfo(
  trio,
  defaults = list(
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  )
)
```

`downloadSubmissionStudy` *Download a BenchmarkStudy from the submission database*

Description

Download a BenchmarkStudy from the submission database

Usage

```
downloadSubmissionStudy(
  studyID,
  name = NULL,
  version = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  cachePath = tempdir()
)
```

Arguments

`studyID`	Study identifier from the `Study` table.
`name`	Optional study name fallback for compatibility. Prefer `studyID` for new code.
`version`	Optional version string used only when loading by `name`.
`ss`	Google Sheets spreadsheet ID containing the submission tables.
`cachePath`	Directory for downloaded files. Defaults to `tempdir()`.

Value

A populated `BenchmarkStudy` object.

Examples

```
if (interactive()) {
  study <- downloadSubmissionStudy(
    studyID =  "ST005",
    cachePath = tempdir()
  )
  study
}
```

downloadSubmissionTrio *Download a Trio from the five-table submission database*

Description

Download a Trio from the five-table submission database

Usage

```
downloadSubmissionTrio(
  datasetID,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  cachePath = tempdir()
)
```

Arguments

datasetID	Dataset identifier from the Dataset table.
ss	Google Sheets spreadsheet ID containing the five database tables.
cachePath	Directory for downloaded files. Defaults to tempdir().

Value

A populated Trio object.

getSubmissionStudy *Get one existing Study row by studyID*

Description

Get one existing Study row by studyID

Usage

```
getSubmissionStudy(
  studyID,
  studies = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg"
)
```

Arguments

studyID	Existing Study identifier.
studies	Optional data frame of Study rows.
ss	Submission spreadsheet ID or URL.

Value

A one-row data frame for the requested Study.

Examples

```
studies <- data.frame(
  studyID = "study_001",
  studyName = "example_study",
  version = "0.0.1",
  description = "A small example benchmark study.",
  type = "original",
  protocolGist = "",
  mappingFunctions = "",
  stringsAsFactors = FALSE
)
getSubmissionStudy(
  "study_001",
  studies = studies
)
```

---

getSubmissionStudyDatasets

Get linked StudyDataset rows by studyID

Description

Get linked StudyDataset rows by studyID

Usage

```
getSubmissionStudyDatasets(
  studyID,
  study_datasets = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg"
)
```

Arguments

`studyID`	Existing Study identifier.
`study_datasets`	Optional data frame of StudyDataset rows.
`ss`	Submission spreadsheet ID or URL.

Value

A data frame of StudyDataset rows linked to the supplied studyID.

Examples

```
study_datasets <- data.frame(
  studyDatasetID = "study_dataset_001",
  studyID = "study_001",
  datasetID = "dataset_001",
  stringsAsFactors = FALSE
)
getSubmissionStudyDatasets(
  "study_001",
  study_datasets = study_datasets
)
```

---

ghCIndexMetric *GH C-Index Metric*

Description

Computes the GH C-Index for survival analysis.

Usage

```
ghCIndexMetric(evidence, predicted)
```

Arguments

evidence	The true survival times and event indicators.
predicted	The predicted survival times.

Value

The GH C-Index.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
  event = c(1, 0, 1, 0))  # Testing
)
# Predicted risk scores
predicted <- list(
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
ghCIndexMetric(evidence, predicted)
```

harrelCIndexMetric *Harrel's C-Index Metric*

Description

Computes Harrel's C-Index for survival analysis.

Usage

```
harrelCIndexMetric(evidence, predicted)
```

Arguments

evidence	The true survival times and event indicators.
predicted	The predicted survival times.

Value

Harrel's C-Index.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
  event = c(1, 0, 1, 0))  # Testing
)
# Predicted risk scores
predicted <- list(
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
harrelCIndexMetric(evidence, predicted)
```

interactivePrepareStudySubmission *Interactively prepare a Study submission*

Description

Interactively prepare a Study submission

Usage

```
interactivePrepareStudySubmission(
  study,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  githubPat = Sys.getenv("GITHUB_PAT"),
  gistPublic = TRUE,
  url = submission_webapp_url,
  review = TRUE,
  build_payload = TRUE,
  build_json = TRUE
)
```

Arguments

study	A BenchmarkStudy object.
ss	Submission spreadsheet ID or URL.
githubPat	Optional GitHub personal access token.
gistPublic	Logical; whether uploaded gists should be public.
url	Google Apps Script endpoint URL. Defaults to the package submission web app.
review	Whether to print a short submission summary. Defaults to TRUE.
build_payload	Whether to include the nested payload. Defaults to TRUE.
build_json	Whether to include JSON output. Defaults to TRUE.

Value

A named list containing the Study submission bundle.

---

interactivePrepareStudyUpdateSubmission

Interactively prepare a Study update submission

Description

Interactively prepare a Study update submission

Usage

```
interactivePrepareStudyUpdateSubmission(
  studyID = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  githubPat = Sys.getenv("GITHUB_PAT"),
  gistPublic = TRUE,
  url = submission_webapp_url,
  review = TRUE,
```

```
  build_payload = TRUE,
  build_json = TRUE
)
```

Arguments

studyID	Optional existing Study identifier to update. When omitted, the user is prompted to choose one.
ss	Submission spreadsheet ID or URL.
githubPat	Optional GitHub personal access token.
gistPublic	Logical; whether uploaded gists should be public.
url	Google Apps Script endpoint URL. Defaults to the package submission web app.
review	Whether to print a short submission summary. Defaults to TRUE.
build_payload	Whether to include the nested payload. Defaults to TRUE.
build_json	Whether to include JSON output. Defaults to TRUE.

Value

A named list containing the updated Study submission bundle.

JSDmetric *Jensen-Shannon Divergence (JSD) Metric*

Description

Computes the Jensen-Shannon divergence between two non-negative numeric vectors after normalizing them to probability distributions.

Usage

```
JSDmetric(evidence, predicted)
```

Arguments

evidence	The true values.
predicted	The predicted values.

Value

The Jensen-Shannon divergence.

Examples

```
evidence <- c(0.2, 0.3, 0.5)
predicted <- c(0.1, 0.4, 0.5)
JSDmetric(evidence, predicted)
```

kdeMetric *Kernel Density Estimation (KDE) Metric*

Description

Computes the kernel density estimation test statistic.

Usage

```
kdeMetric(evidence, predicted)
```

Arguments

evidence	The true values.
predicted	The predicted values.

Value

The KDE test statistic.

Examples

```
evidence <- c(1, 2, 3, 4)
predicted <- c(1.1, 2.1, 2.9, 4.2)
kdeMetric(evidence, predicted)
```

listCuratedTrioDatasets *List the curated Trio datasets*

Description

List the curated Trio datasets

Usage

```
listCuratedTrioDatasets(name_filter = NULL, dataType_filter = NULL)
```

Arguments

name_filter	A string to filter datasets by name (case-insensitive partial match)
dataType_filter	A string or vector of strings to filter datasets by data type

Value

A data frame with the dataset names and IDs.

listCuratedTrioStudies *List the curated Trio studies*

Description

List the curated Trio studies

Usage

```
listCuratedTrioStudies(name_filter = NULL)
```

Arguments

name_filter	A string to filter studies by name (case-insensitive partial match)

Value

A data frame with the study names and IDs.

listSubmissionStudies *List existing Study rows*

Description

List existing Study rows

Usage

```
listSubmissionStudies(ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg")
```

Arguments

ss	Submission spreadsheet ID or URL.

Value

A data frame of existing Study rows.

listSubmissionStudyDatasets

List available datasets for Study submission

Description

List available datasets for Study submission

Usage

```
listSubmissionStudyDatasets(
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg"
)
```

Arguments

ss	Submission spreadsheet ID or URL.

Value

A data frame of existing Dataset rows.

lubomski_microbiome_data

Human Gut Microbiome Abundance and Patient Classes

Description

Data set consists of a matrix of abundances of 1192 microbial taxa for 575 samples and a factor vector of classes for Parkinson's disease for the same 575 patients

Format

x has a row for each sample and a column for each taxon. lubomPD is a factor vector with values 0 representing Healthy Control (HC) and 1 representing Parkinson's Disease (PD).

Source

An R package PD16Sdata, *BMC*, 2023. Webpage: https://microbiomejournal.biomedcentral.com/articles/10.1186/s40168-023-01475-4

macroF1Metric *Macro F1 Score Metric*

Description

Computes the macro F1 score of the predictions.

Usage

```
macroF1Metric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The macro F1 score.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
macroF1Metric(evidence, predicted)
```

macroPrecMetric *Macro Precision Metric*

Description

Computes the macro precision of the predictions.

Usage

```
macroPrecMetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The macro precision.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
macroPrecMetric(evidence, predicted)
```

---

macroRecMetric *Macro Recall Metric*

Description

Computes the macro recall of the predictions.

Usage

```
macroRecMetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The macro recall.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
macroRecMetric(evidence, predicted)
```

MCCmetric *Matthews Correlation Coefficient (MCC) Metric*

Description

Computes the Matthews Correlation Coefficient (MCC) of the predictions.

Usage

```
MCCmetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The MCC.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
MCCmetric(evidence, predicted)
```

`microF1Metric` *Micro F1 Score Metric*

Description

Computes the micro F1 score of the predictions.

Usage

```
microF1Metric(evidence, predicted)
```

Arguments

`evidence`	The true labels.
`predicted`	The predicted labels.

Value

The micro F1 score.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
microF1Metric(evidence, predicted)
```

microPrecMetric *Micro Precision Metric*

Description

Computes the micro precision of the predictions.

Usage

```
microPrecMetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The micro precision.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
microPrecMetric(evidence, predicted)
```

microRecMetric *Micro Recall Metric*

Description

Computes the micro recall of the predictions.

Usage

```
microRecMetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The micro recall.

Examples

```
evidence <- factor(c("A", "B", "A", "B"))
predicted <- factor(c("A", "A", "A", "B"))
microRecMetric(evidence, predicted)
```

MSEmetric *Mean Squared Error (MSE) Metric*

Description

Computes the mean squared error of the predictions.

Usage

```
MSEmetric(evidence, predicted)
```

Arguments

evidence	The true values.
predicted	The predicted values.

Value

The mean squared error.

Examples

```
evidence <- c(1, 2, 3, 4)
predicted <- c(1.1, 2.1, 2.9, 4.2)
MSEmetric(evidence, predicted)
```

NMImetric *Normalized Mutual Information (NMI) Metric*

Description

Computes the normalized mutual information between two cluster labelings.

Usage

```
NMImetric(evidence, predicted)
```

Arguments

evidence	The true labels.
predicted	The predicted labels.

Value

The normalized mutual information.

Examples

```
evidence <- factor(c("A", "A", "B", "B"))
predicted <- factor(c("A", "A", "B", "B"))
NMImetric(evidence, predicted)
```

`prepareStudySubmission` *Prepare a Study submission bundle*

Description

Prepare a Study submission bundle

Usage

```
prepareStudySubmission(
  study,
  datasetIDs = NULL,
  available_datasets = NULL,
  existing_studies = NULL,
  defaults = list(),
  protocolGist = NULL,
  mappingFunctions = NULL,
  protocolFile = NULL,
  mappingFunctionsFile = NULL,
  uploadProtocol = FALSE,
  uploadMappingFunctions = FALSE,
  githubPat = Sys.getenv("GITHUB_PAT"),
  gistPublic = TRUE,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  build_payload = TRUE,
  build_json = TRUE,
  review = TRUE,
  submit = FALSE,
  url = submission_webapp_url,
  submittedBy = NULL
)
```

Arguments

`study`	A `BenchmarkStudy` object.
`datasetIDs`	Optional character vector of existing dataset IDs to link.

`available_datasets`
: Optional data frame of Dataset rows for selection.

`existing_studies`
: Optional data frame of Study rows.

`defaults`
: Optional named list for `collectStudySubmissionInfo()`.

`protocolGist`
: Optional protocol gist URL.

`mappingFunctions`
: Optional mapping functions gist URL.

`protocolFile`
: Optional local file path to upload as a protocol gist.

`mappingFunctionsFile`
: Optional local file path to upload as a mapping functions gist.

`uploadProtocol`
: Logical; whether to upload `protocolFile` to a gist.

`uploadMappingFunctions`
: Logical; whether to upload `mappingFunctionsFile` to a gist.

`githubPat`
: Optional GitHub personal access token.

`gistPublic`
: Logical; whether uploaded gists should be public.

`ss`
: Submission spreadsheet ID or URL.

`build_payload`
: Whether to include the nested payload. Defaults to `TRUE`.

`build_json`
: Whether to include JSON output. Defaults to `TRUE`.

`review`
: Whether to print a short submission summary. Defaults to `TRUE`.

`submit`
: Whether to submit immediately. Defaults to `FALSE`.

`url`
: Google Apps Script endpoint URL. Defaults to the package submission web app.

`submittedBy`
: Submitter email or identifier.

Value

A named list containing the Study submission bundle.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
available_datasets <- data.frame(
  datasetID = "dataset_001",
  name = "example_dataset",
  stringsAsFactors = FALSE
)
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
```

```
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
result <- prepareStudySubmission(
  study,
  datasetIDs = "dataset_001",
  available_datasets = available_datasets,
  existing_studies = existing_studies,
  build_json = FALSE,
  review = FALSE
)
names(result)
```

`prepareStudyUpdateSubmission` *Prepare an update submission from an existing Study version*

Description

Prepare an update submission from an existing Study version

Usage

```
prepareStudyUpdateSubmission(
  studyID,
  study = NULL,
  datasetIDs = NULL,
  description = NULL,
  protocolGist = NULL,
  mappingFunctions = NULL,
  studies = NULL,
  study_datasets = NULL,
  ss = "1H8hOxL8D0XTquao8vGZ2cr9-XeaFC48SWAdFn0M3fkg",
  build_payload = TRUE,
  build_json = TRUE,
  review = TRUE,
  submit = FALSE,
  url = submission_webapp_url,
  submittedBy = NULL
)
```

Arguments

`studyID`	Existing Study identifier to use as the update baseline.
`study`	Optional `BenchmarkStudy` object. When omitted, a new one is created from the existing Study row.
`datasetIDs`	Optional replacement dataset IDs. Defaults to the linked datasets from the baseline Study.

description	Optional replacement description.
protocolGist	Optional replacement protocol gist URL.
mappingFunctions	Optional replacement mapping functions gist URL.
studies	Optional data frame of Study rows.
study_datasets	Optional data frame of StudyDataset rows.
ss	Submission spreadsheet ID or URL.
build_payload	Whether to include the nested payload. Defaults to TRUE.
build_json	Whether to include JSON output. Defaults to TRUE.
review	Whether to print a short submission summary. Defaults to TRUE.
submit	Whether to submit immediately. Defaults to FALSE.
url	Google Apps Script endpoint URL. Defaults to the package submission web app.
submittedBy	Submitter email or identifier.

Value

A named list containing the updated Study submission bundle.

Examples

```
studies <- data.frame(
  studyID = "study_001",
  studyName = "example_study",
  version = "0.0.1",
  description = "A small example benchmark study.",
  type = "original",
  protocolGist = "",
  mappingFunctions = "",
  stringsAsFactors = FALSE
)
study_datasets <- data.frame(
  studyDatasetID = "study_dataset_001",
  studyID = "study_001",
  datasetID = "dataset_001",
  stringsAsFactors = FALSE
)
result <- prepareStudyUpdateSubmission(
  studyID = "study_001",
  studies = studies,
  study_datasets = study_datasets,
  build_json = FALSE,
  review = FALSE
)
names(result)
```

`prepareTrioSubmissionBundle` *Build a pre-submit Trio submission bundle*

Description

This is the higher-level R interface for the new submission workflow. It can prepare dataset/evidence files, prepare metric metadata, build the five-table submission object, and return the payload/JSON for inspection before calling `submitTrioSubmission()`.

Usage

```
prepareTrioSubmissionBundle(
  trio,
  dataset_args,
  task_args,
  evidence_task_map,
  prepare_files = FALSE,
  file_args = list(),
  prepare_metrics = FALSE,
  metric_args = list(),
  build_payload = TRUE,
  build_json = FALSE
)
```

Arguments

`trio`	A `Trio` object.
`dataset_args`	Named list passed to `buildDatasetSubmission()`.
`task_args`	Named list passed to `buildDatasetTaskSubmission()`.
`evidence_task_map`	Named character vector mapping Trio evidence names to submission task names.
`prepare_files`	Logical; if `TRUE`, call `prepareTrioSubmissionFiles()`.
`file_args`	Named list of additional arguments for `prepareTrioSubmissionFiles()`, excluding `trio`.
`prepare_metrics`	Logical; if `TRUE`, call `prepareTrioSubmissionMetrics()`.
`metric_args`	Named list of additional arguments for `prepareTrioSubmissionMetrics()`, excluding `trio`.
`build_payload`	Logical; if `TRUE`, attach the nested payload structure.
`build_json`	Logical; if `TRUE`, attach the JSON string.

Value

A named list containing the built `submission`, plus optional `files`, `metrics`, `payload`, and `json`.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
dataset_args <- list(
  dataType = "omics",
  dataModality = "transcriptomics",
  technology = "RNA-seq",
  tissue = "blood",
  status = "healthy"
)
task_args <- list(
  taskStage = "prediction",
  taskType = "classification",
  taskName = "class_prediction"
)
bundle <- prepareTrioSubmissionBundle(
  trio = trio,
  dataset_args = dataset_args,
  task_args = task_args,
  evidence_task_map = c(class_labels = "class_prediction")
)
names(bundle)
```

`prepareTrioSubmissionFiles` *Prepare Trio files and download metadata for submission*

Description

This helper reuses the good parts of the old `writeCTD()` workflow for the new submission path. It can either save dataset/evidence files locally so the user can upload them, or reuse an existing downloadable source already attached to the `Trio`.

Usage

```
prepareTrioSubmissionFiles(
  trio,
  name = trio$name,
```

```
  outputDir = ".",
  saveData = NULL,
  saveEvidence = NULL,
  useExistingSource = NULL,
  figshareUrl = NULL,
  datasetFileName = NULL,
  evidenceFileName = NULL,
  verifyFigshare = FALSE,
  skipMd5Check = FALSE
)
```

Arguments

`trio`	A `Trio` object.
`name`	Dataset name used for generated filenames. Defaults to `trio$name`.
`outputDir`	Directory for generated `.rds` files. Defaults to `"."`.
`saveData`	Logical indicating whether to save `trio$data` to an RDS file. If `NULL` and interactive, the user is prompted.
`saveEvidence`	Logical indicating whether to save all supporting evidence to a single RDS file. If `NULL` and interactive, defaults to `TRUE`.
`useExistingSource`	Logical indicating whether to keep using `trio$dataSource`/`trio$dataSourceID` rather than preparing uploaded files. If `NULL` and interactive, the user is prompted when an existing source is available.
`figshareUrl`	Optional Figshare URL to verify against prepared files.
`datasetFileName`	Optional dataset filename to match inside the Figshare article.
`evidenceFileName`	Optional evidence filename to match inside the Figshare article.
`verifyFigshare`	Logical; if `TRUE`, verify the prepared files against the Figshare article metadata.
`skipMd5Check`	Logical; if `TRUE`, allow verification without MD5 matching.

Value

A named list describing the saved files and/or reusable download sources for dataset and evidence.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
```

```
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
files <- prepareTrioSubmissionFiles(
  trio,
  outputDir = tempdir(),
  saveData = TRUE,
  saveEvidence = TRUE,
  useExistingSource = FALSE
)
names(files)
```

`prepareTrioSubmissionMetrics` *Prepare Trio metric metadata for submission*

Description

This helper aligns the new submission workflow with the older `writeCTD()` metric handling. Internal package metrics are recorded directly. Custom metrics can optionally be uploaded to a GitHub gist and linked from the resulting `Metric` table rows.

Usage

```
prepareTrioSubmissionMetrics(
  trio,
  name = trio$name,
  uploadCustom = FALSE,
  githubPat = NULL,
  gistPublic = TRUE
)
```

Arguments

`trio`	A `Trio` object.
`name`	Dataset name used for the gist filename. Defaults to `trio$name`.
`uploadCustom`	Logical; if `TRUE`, custom metrics are uploaded to a GitHub gist.
`githubPat`	Optional GitHub personal access token with gist scope.
`gistPublic`	Logical; whether the created gist should be public.

Value

A list with entries `Metric`, `gist`, and `custom_metric_lines`.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
prepareTrioSubmissionMetrics(trio)
```

RMSEmetric *Root Mean Squared Error (RMSE) Metric*

Description

Computes the root mean squared error of the predictions.

Usage

```
RMSEmetric(evidence, predicted)
```

Arguments

evidence	The true values.
predicted	The predicted values.

Value

The root mean squared error.

Examples

```
evidence <- c(1, 2, 3, 4)
predicted <- c(1.1, 2.1, 2.9, 4.2)
RMSEmetric(evidence, predicted)
```

`studySubmissionToJSON` *Convert a Study submission to JSON*

Description

Convert a Study submission to JSON

Usage

```
studySubmissionToJSON(submission, pretty = TRUE)
```

Arguments

`submission`	A submission object returned by `buildStudySubmission()`.
`pretty`	Whether to pretty-print the JSON. Defaults to `TRUE`.

Value

A JSON string.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
submission <- buildStudySubmission(
  study,
  datasetIDs = "dataset_001",
  existing_studies = existing_studies
)
json <- studySubmissionToJSON(submission)
substr(json, 1, 20)
```

`submitStudySubmission` *Submit a Study submission payload to Google Apps Script*

Description

Submit a Study submission payload to Google Apps Script

Usage

```
submitStudySubmission(submission, url, submittedBy)
```

Arguments

`submission`	A submission object returned by `buildStudySubmission()`.
`url`	Google Apps Script endpoint URL.
`submittedBy`	Submitter email or identifier.

Value

A list containing request status information and response text.

Examples

```
study <- BenchmarkStudy$new(name = "example_study")
study$description <- "A small example benchmark study."
existing_studies <- data.frame(
  studyID = character(0),
  studyName = character(0),
  version = character(0),
  description = character(0),
  type = character(0),
  protocolGist = character(0),
  mappingFunctions = character(0),
  stringsAsFactors = FALSE
)
submission <- buildStudySubmission(
  study,
  datasetIDs = "dataset_001",
  existing_studies = existing_studies
)

if (interactive() && curl::has_internet()) {
  response <- submitStudySubmission(
    submission,
    url = "https://script.google.com/macros/s/example/exec",
    submittedBy = "researcher@example.org"
  )
  names(response)
}
```

submitTrioSubmission *Submit a Trio submission payload to Google Apps Script*

Description

This helper posts the Trio submission payload to a Google Apps Script endpoint using the same redirect-tolerant approach as the prototype script.

Usage

```
submitTrioSubmission(submission, url, submittedBy, submittedType = "Trio")
```

Arguments

submission	A submission object returned by buildTrioSubmission().
url	Google Apps Script endpoint URL.
submittedBy	Submitter email or identifier.
submittedType	Submission type label. Defaults to "Trio".

Value

A list containing request status information and response text.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
submission <- buildTrioSubmission(
  trio = trio,
  dataset_args = list(
    dataType = "omics",
    dataModality = "transcriptomics",
    technology = "RNA-seq",
    tissue = "blood",
    status = "healthy"
  ),
  task_args = list(
```

```
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  ),
  evidence_task_map = c(class_labels = "class_prediction")
)

if (interactive() && curl::has_internet()) {
  response <- submitTrioSubmission(
    submission,
    url = "https://script.google.com/macros/s/example/exec",
    submittedBy = "researcher@example.org"
  )
  names(response)
}
```

`timeDependentAUCMetric` *Time-Dependent AUC Metric*

Description

Computes the time-dependent AUC for survival analysis.

Usage

```
timeDependentAUCMetric(evidence, predicted)
```

Arguments

`evidence`	The true survival times and event indicators.
`predicted`	The predicted survival times.

Value

The time-dependent AUC.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
  event = c(1, 0, 1, 0))  # Testing
)

# Predicted risk scores
predicted <- list(
```

```
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
timeDependentAUCMetric(evidence, predicted)
```

`Trio` *A Trio object*

Description

An object containing a dataset and methods for evaluating analytical tasks against ground truths for the dataset.

Value

A `Trio` object

Public fields

`data` The data

`evidence` The supporting evidence for the data

`metrics` The metric for evaluating tasks against the gold standards

`cachePath` The path to the data cache

`dataSource` The data repository that the data were retrieved from

`dataSourceID` The dataset ID for `dataSource`

`evidenceSource` The data repository that the supporting evidence was retrieved from.

`evidenceSourceID` The dataset ID for `evidenceSource`.

`splitIndices` Indices for cross-validation

`splitSeed` The seed used to generate the split indices

`verbose` Set the verbosity of Trio. Defaults to `FALSE`.

`description` A description of the dataset.

`name` The name of the Trio object, as defined in Curated Trio Datasets.

Methods

Public methods:

- `Trio$new()`
- `Trio$addEvidence()`
- `Trio$addMetric()`
- `Trio$getMetrics()`
- `Trio$getEvidence()`
- `Trio$evaluate()`

- Trio$split()
- Trio$print()
- Trio$writeCTD()
- Trio$clone()

Method `new()`**:** Create a Trio object

Usage:

```
Trio$new(
  datasetID = NULL,
  data = NULL,
  dataLoader = NULL,
  evidenceID = NULL,
  evidence = NULL,
  evidenceColumns = NULL,
  evidenceLoader = NULL,
  task = NULL,
  metrics = NULL,
  cachePath = FALSE,
  verbose = FALSE,
  description = NULL,
  name = NULL
)
```

Arguments:

`datasetID` A string specifying a dataset, either a name from curated-trio-data or a format string of the form `source:source_id`.

`data` An object to use as the Trio dataset.

`dataLoader` A custom loading function that takes the path of a downloaded file and returns a single dataset, ready to be used in evaluation tasks.

`evidenceID` If `datasetID` is not an ID from the curated trio datasets spreadsheet, then a format string of the form `source:source_id` indicating the file to obtain the supporting evidence from.

`evidence` A named list of lists. The top-level list is named by task type. The lower-level list is of length-two and named `"evidence"` and `"metrics"`. The `"evidence"` component has supporting evidence and the `"metrics"` component has a character vector of metric names (corresponding to the names of the list provided to the `metrics` parameter).

`evidenceColumns` If specified, extract supporting evidence from columns in the loaded dataset.

`evidenceLoader` Alternative to `evidence` and `evidenceColumns`. Extract the evidence in a flexible way.

`task` If `evidenceColumns` or `evidenceLoader` specified, a character vector of length 1 naming the task the evidence is for.

`metrics` A named list of metric functions.

`cachePath` The path to the data cache

`verbose` Set the verbosity of Trio. Defaults to `FALSE`.

`description` A description of the dataset.

`name` The name of the Trio object, as defined in Curated Trio Datasets.

Method `addEvidence()`**:** Add supporting evidence to the Trio.

Usage:

```
Trio$addEvidence(name, evidence, metrics, args = NULL)
```

Arguments:

`name` A string specifying the name of the supporting evidence.

`evidence` The supporting evidence. An object to be compared or a function to be run on the data.

`metrics` A list of one or more metrics names used to compare gs with the input to evaluate.

`args` A named list of parameters and values to be passed to the function.

Method `addMetric()`**:** Add a metric to the Trio.

Usage:

```
Trio$addMetric(name, metric, args = NULL)
```

Arguments:

`name` A string specifying the name of the metric.

`metric` The metric. A function to be run on the input to evaluate to compare it with the gold standard. Should be of the form f(x, y, ...). Where `x` is the "truth" and `y` is the output to be evaluated. Otherwise input a wrapper function of the desired metric.

`args` A named list of parameters and values to be passed to the function.

Method `getMetrics()`**:** Get metrics by supporting evidence name.

Usage:

```
Trio$getMetrics(evidenceName)
```

Arguments:

`evidenceName` A string specifying the name of the supporting evidence.

Method `getEvidence()`**:** Get supporting evidence by name.

Usage:

```
Trio$getEvidence(name)
```

Arguments:

`name` A string specifying the name of the supporting evidence.

Method `evaluate()`**:** Evaluate against gold standards

Usage:

```
Trio$evaluate(input)
```

Arguments:

`input` A named list of objects to be evaluated against gold standards.

Method `split()`**:** Create cross-validation indices.

Usage:

```
Trio$split(
  y,
  n_fold = 5L,
  n_repeat = 1L,
  stratify = TRUE,
  seed = 23624482,
  overwrite = FALSE,
  ...
)
```

Arguments:

`y` A variable to use for stratified sampling (e.g. supporting evidence). If `stratify` is false, a vector the length of the data.

`n_fold` Number of folds. Defaults to `5L`.

`n_repeat` Number of repeats. Defaults to `1L`.

`stratify` If `TRUE`, uses stratified sampling. Defaults to `TRUE`.

`seed` An `integer` of lenth 1. Defaults to 23624482, which is the text "BenchHub" in vanity number form.

`overwrite` If `TRUE`, overwrites the current split. Defaults to `FALSE`.

`...` Additional arguments passed to `splitTools::create_folds`.

Method `print()`**:** Print method to display key information about the Trio object.

Usage:

```
Trio$print()
```

Method `writeCTD()`**:** Write the Trio Metadata to Curated Trio Datasets sheet.

Usage:

```
Trio$writeCTD(
  name,
  email = NULL,
  githubPat = NULL,
  description = NULL,
  figshareUrl = NULL,
  datasetFileName = NULL,
  evidenceFileName = NULL,
  dataType = NULL,
  skipMd5Check = FALSE
)
```

Arguments:

`name` The name of the dataset to be added.

`email` Required. Email address of the contributor for dataset update notifications.

`githubPat` Optional GitHub Personal Access Token. If not provided and not set in environment, will prompt user.

`description` Optional description of the dataset. If not provided and not set, will prompt user.

`figshareUrl` Optional URL to the Figshare dataset. If not provided, will prompt user.

`datasetFileName` Optional name of the dataset file in Figshare. If not provided, will prompt user for selection.

`evidenceFileName` Optional name of the evidence file in Figshare. If not provided, will prompt user for selection.

`dataType` Optional type of data. Must be one of: "omics", "clinical", "spatial", "other". If not provided, will prompt user.

`skipMd5Check` Optional boolean to skip MD5 verification. Defaults to FALSE.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
trio <- Trio$new("figshare:26054188/47112109", cachePath = tempdir())
```

`trioSubmissionToJSON` *Convert a combined Trio submission to JSON*

Description

Convert a combined Trio submission to JSON

Usage

```
trioSubmissionToJSON(submission, pretty = TRUE)
```

Arguments

`submission`	A submission object returned by `buildTrioSubmission()`.
`pretty`	Whether to pretty-print the JSON. Defaults to `TRUE`.

Value

A JSON string.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
```

```
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
submission <- buildTrioSubmission(
  trio = trio,
  dataset_args = list(
    dataType = "omics",
    dataModality = "transcriptomics",
    technology = "RNA-seq",
    tissue = "blood",
    status = "healthy"
  ),
  task_args = list(
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  ),
  evidence_task_map = c(class_labels = "class_prediction")
)
json <- trioSubmissionToJSON(submission)
substr(json, 1, 20)
```

`unoCIndexMetric` *Uno's C-Index Metric*

Description

Computes Uno's C-Index for survival analysis.

Usage

```
unoCIndexMetric(evidence, predicted)
```

Arguments

`evidence`	The true survival times and event indicators.
`predicted`	The predicted survival times.

Value

Uno's C-Index.

Examples

```
# More realistic training dataset (8 patients)
evidence <- list(
  survival::Surv(time = c(5, 10, 15, 20, 25, 30, 35, 40),
  event = c(1, 1, 0, 1, 0, 1, 1, 0)),  # Training
  survival::Surv(time = c(12, 18, 25, 32),
```

```
  event = c(1, 0, 1, 0))  # Testing
)
# Predicted risk scores
predicted <- list(
  c(0.5142118, 0.3902035, 0.9057381, 0.4469696,
  0.8360043, 0.7375956, 0.8110551, 0.3881083),  # Training predictions
  c(0.685169729, 0.003948339, 0.832916080, 0.007334147)  # Testing predictions
)
unoCIndexMetric(evidence, predicted)
```

`writeSubmission` *Interactively prepare a Trio submission bundle*

Description

This high-level helper provides a `writeCTD()`-style console workflow for the new five-table submission path. It collects dataset, task, evidence, and metric metadata; optionally prepares local files for upload and verifies a Figshare article; optionally uploads custom metrics to a GitHub gist; and returns all submission tables ready for review before calling `submitTrioSubmission()`.

Usage

```
writeSubmission(
  trio,
  n_tasks = NULL,
  dataset_defaults = list(),
  task_defaults = list(),
  evidence_defaults = list(),
  metric_defaults = list(),
  prepare_files = TRUE,
  file_args = list(),
  upload_custom_metrics = FALSE,
  githubPat = Sys.getenv("GITHUB_PAT"),
  gistPublic = TRUE,
  review = TRUE,
  submit = NULL,
  url = submission_webapp_url,
  submittedBy = NULL,
  build_payload = TRUE,
  build_json = FALSE
)
```

Arguments

`trio`	A `Trio` object.
`n_tasks`	Optional number of tasks to define. If `NULL`, interactive sessions prompt for it and non-interactive sessions require it through `task_defaults` or `n_tasks`.

dataset_defaults
: Optional named list passed into `collectDatasetSubmissionInfo()`.

task_defaults
: Optional named list passed into `collectTaskSubmissionInfo()`.

evidence_defaults
: Optional named list passed into `collectEvidenceSubmissionInfo()`.

metric_defaults
: Optional named list passed into `collectMetricSubmissionInfo()`.

prepare_files
: Logical; if `TRUE`, prepare dataset/evidence files or reuse an existing source.

file_args
: Optional named list passed into `prepareTrioSubmissionFiles()`.

upload_custom_metrics
: Logical; if `TRUE`, upload custom metrics to a GitHub gist.

githubPat
: Optional GitHub personal access token. Defaults to the current `GITHUB_PAT` environment variable.

gistPublic
: Logical; whether any created gist should be public.

review
: Logical; if `TRUE`, print a compact review summary before returning.

submit
: Logical; if `TRUE`, submit the built payload at the end. If `NULL`, interactive sessions ask whether to submit after the review step.

url
: Optional Google Apps Script endpoint URL used when `submit = TRUE`.

submittedBy
: Optional submitter email or identifier used when `submit = TRUE`.

build_payload
: Logical; if `TRUE`, attach the nested payload object.

build_json
: Logical; if `TRUE`, attach the JSON string.

Value

A named list containing collected arguments, optional prepared file and metric metadata, the final `submission`, and optional `payload`/`json`.

Examples

```
data <- data.frame(feature = c(1, 2, 3), row.names = paste0("sample", 1:3))
labels <- factor(c("A", "B", "A"))
names(labels) <- rownames(data)
trio <- Trio$new(
  data = data,
  evidence = list(class_labels = list(
    evidence = labels,
    metrics = "macroF1Metric"
  )),
  metrics = list(macroF1Metric = macroF1Metric),
  name = "example_dataset",
  description = "A small example dataset."
)
result <- writeSubmission(
  trio = trio,
  n_tasks = 1,
  dataset_defaults = list(
```

```
    dataType = "omics",
    dataModality = "transcriptomics",
    technology = "RNA-seq",
    tissue = "blood",
    status = "healthy"
  ),
  task_defaults = list(
    taskStage = "prediction",
    taskType = "classification",
    taskName = "class_prediction"
  ),
  evidence_defaults = list(
    taskName = "class_prediction",
    evidenceType = "experimental_ground_truth"
  ),
  metric_defaults = list(metricType = "label_based"),
  prepare_files = FALSE,
  build_json = FALSE,
  review = FALSE
)
names(result)
```

Index

∗ **datasets**
lubomski_microbiome_data, 38

ARImetric, 3

balAccMetric, 4
balErrMetric, 4
beggCIndexMetric, 5
BenchmarkInsights, 6
BenchmarkStudy, 8
brierScoreMetric, 12
buildDatasetEvidenceSubmission, 12
buildDatasetSubmission, 13
buildDatasetTaskMetricSubmission, 15
buildDatasetTaskSubmission, 16
buildMetricSubmission, 17
buildStudySubmission, 18
buildStudySubmissionPayload, 19
buildTrioSubmission, 20
buildTrioSubmissionPayload, 21

collectDatasetSubmissionInfo, 22
collectDatasetTaskMetricSubmission, 23
collectEvidenceSubmissionInfo, 24
collectMetricSubmissionInfo, 25
collectStudySubmissionInfo, 26
collectTaskSubmissionInfo, 28

downloadSubmissionStudy, 29
downloadSubmissionTrio, 30

getSubmissionStudy, 30
getSubmissionStudyDatasets, 31
ghCIndexMetric, 32

harrelCIndexMetric, 33

interactivePrepareStudySubmission, 33
interactivePrepareStudyUpdateSubmission, 34

JSDmetric, 35

kdeMetric, 36

listCuratedTrioDatasets, 36
listCuratedTrioStudies, 37
listSubmissionStudies, 37
listSubmissionStudyDatasets, 38
lubomPD *(*lubomski_microbiome_data*)*, 38
lubomski_microbiome_data, 38

macroF1Metric, 39
macroPrecMetric, 39
macroRecMetric, 40
MCCmetric, 40
microF1Metric, 41
microPrecMetric, 42
microRecMetric, 42
MSEmetric, 43

NMImetric, 43

prepareStudySubmission, 44
prepareStudyUpdateSubmission, 46
prepareTrioSubmissionBundle, 48
prepareTrioSubmissionFiles, 49
prepareTrioSubmissionMetrics, 51

RMSEmetric, 52

studySubmissionToJSON, 53
submitStudySubmission, 54
submitTrioSubmission, 55

timeDependentAUCMetric, 56
Trio, 57
trioSubmissionToJSON, 61

unoCIndexMetric, 62

writeSubmission, 63

x *(*lubomski_microbiome_data*)*, 38