Package 'cometr'

October 19, 2023

Title 'Comet' API for R

Version 0.4.0

Description A convenient 'R' wrapper to the 'Comet' API, which is a cloud platform allowing you to track, compare, explain and optimize machine learning experiments and models. Experiments can be viewed on the 'Comet' online dashboard at <https://www.comet.com>.

URL https://github.com/comet-ml/cometr

BugReports https://github.com/comet-ml/cometr/issues

Imports callr, httr, jsonlite, R.utils, R6 (>= 2.4.0), utils, yaml, digest

Suggests covr, curl, git2r (>= 0.22.1), httptest, ps, testthat, withr

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Dean Attali [aut],
Doug Blank [aut],
Iaroslav Omelianenko [aut] (<https://orcid.org/0000-0002-2190-5664>),
Nimrod Lahav [cre],
Comet ML, Inc. [cph, fnd]

Maintainer Nimrod Lahav <nimrod@comet.com>

Repository CRAN

Date/Publication 2023-10-19 12:10:08 UTC

R topics documented:

Artifact 2
ArtifactAsset 5
call_api 7

create_artifact 8
create_experiment 9
create_project 10
delete_project 11
disable_logging 12
Experiment 12
get_api_version 22
get_columns 22
get_experiment 23
get_experiments 24
get_multi_metric_chart 25
get_projects 26
get_workspaces 27
LoggedArtifact 27
LoggedArtifactAsset 31

Index **34**

`Artifact` *A Comet Artifact object*

Description

Comet Artifacts allow keeping track of assets beyond any particular experiment. You can keep track of Artifact versions, create many types of assets, manage them, and use them in any step in your ML pipelines - from training to production deployment.

Artifacts live in a Comet Project, are identified by their name and version string number.

Methods

Public methods:

- `Artifact$new()`
- `Artifact$get_artifact_name()`
- `Artifact$get_artifact_type()`
- `Artifact$get_artifact_version()`
- `Artifact$get_aliases()`
- `Artifact$get_metadata()`
- `Artifact$get_version_tags()`
- `Artifact$get_assets()`
- `Artifact$add()`
- `Artifact$add_remote()`
- `Artifact$add_asset()`

Method `new()`**:** Creates new `Artifact` object with provided parameters. After that, the `Artifact` object can be used to save assets and can be logged with an `Experiment`.

Usage:

```
Artifact$new(
  artifact_name,
  artifact_type,
  artifact_version = NULL,
  aliases = NULL,
  metadata = NULL,
  version_tags = NULL
)
```

Arguments:

`artifact_name` (Required) Artifact name.

`artifact_type` (Required) The artifact type, for example 'dataset'.

`artifact_version` The version number to create. If not provided, a new version number will be created automatically.

`aliases` List of aliases. Some aliases to attach to the future Artifact Version. The aliases list is normalized to remove duplicates.

`metadata` Some additional meta-data to attach to the future Artifact Version.

`version_tags` List of tags to be attached to the future Artifact Version.

Method `get_artifact_name()`**:** Get the name of the artifact.

Usage:

```
Artifact$get_artifact_name()
```

Method `get_artifact_type()`**:** Get the type of the artifact.

Usage:

```
Artifact$get_artifact_type()
```

Method `get_artifact_version()`**:** Get the version of the artifact.

Usage:

```
Artifact$get_artifact_version()
```

Method `get_aliases()`**:** Get the version of the artifact.

Usage:

```
Artifact$get_aliases()
```

Method `get_metadata()`**:** Get the metadata of the artifact.

Usage:

```
Artifact$get_metadata()
```

Method `get_version_tags()`**:** Get the list of tags of the artifact version.

Usage:

```
Artifact$get_version_tags()
```

Method `get_assets()`**:** Get the list of assets of the artifact version.

Usage:

```
Artifact$get_assets()
```

Method `add()`**:** Add a local asset to the current pending artifact object.

Usage:

```
Artifact$add(
  local_path,
  overwrite = FALSE,
  logical_path = NULL,
  metadata = NULL
)
```

Arguments:

`local_path` (Required) Either a file/directory path of the files you want to log

`overwrite` If `TRUE` will overwrite all existing assets with the same name.

`logical_path` A custom file name to be displayed. If not provided the file name from the `local_path` argument will be used.

`metadata` Some additional data to attach to the asset.

Method `add_remote()`**:** Add a remote asset to the current pending artifact object. A Remote Asset is an asset but its content is not uploaded and stored on Comet. Rather a link for its location is stored so you can identify and distinguish between two experiment using different version of a dataset stored somewhere else.

Usage:

```
Artifact$add_remote(
  uri,
  logical_path = NULL,
  overwrite = FALSE,
  metadata = NULL
)
```

Arguments:

`uri` (Required) The remote asset location, there is no imposed format and it could be a private link.

`logical_path` The "name" of the remote asset, could be a dataset name, a model file name.

`overwrite` If `TRUE` will overwrite all existing assets with the same name.

`metadata` Some additional data to attach to the asset.

Method `add_asset()`**:** Adds an initialized `ArtifactAsset` object to this Artifact.

Usage:

```
Artifact$add_asset(asset)
```

Arguments:

`asset` The initialized `ArtifactAsset` object

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()
```

```
# Create a Comet Artifact
artifact <- Artifact$new(artifact_name = "Artifact-Name", artifact_type = "Artifact-Type")
artifact$add("local-file")

exp$log_artifact(artifact)
exp$stop()

## End(Not run)
```

`ArtifactAsset` *An Artifact Asset object*

Description

The `ArtifactAsset` represent local or remote asset added to an `Artifact` object but not yet uploaded

Methods

Public methods:

- `ArtifactAsset$new()`
- `ArtifactAsset$get_local_path()`
- `ArtifactAsset$get_logical_path()`
- `ArtifactAsset$is_remote()`
- `ArtifactAsset$has_overwrite()`
- `ArtifactAsset$get_size()`
- `ArtifactAsset$get_link()`
- `ArtifactAsset$get_metadata()`
- `ArtifactAsset$get_asset_type()`

Method `new()`**:** Creates a new `ArtifactAsset` object with provided parameters.

Usage:

```
ArtifactAsset$new(
  logical_path,
  overwrite = FALSE,
  remote = FALSE,
  size = 0,
  link = NULL,
  local_path = NULL,
  metadata = NULL,
  asset_type = NULL
)
```

Arguments:

`logical_path` the logical file name.
`overwrite` If `TRUE` will overwrite all existing assets with the same name.
`remote` Is the asset a remote asset or not.
`size` The size if the asset of a non-remote asset.
`link` The remote link if the asset is remote.
`local_path` The local file path if the asset is non-remote.
`metadata` The metadata to be associated with the asset.
`asset_type` The type of asset.

Method `get_local_path()`**:** Asset local path if the asset is non-remote

Usage:

```
ArtifactAsset$get_local_path()
```

Method `get_logical_path()`**:** Asset logical file name

Usage:

```
ArtifactAsset$get_logical_path()
```

Method `is_remote()`**:** Is the asset a remote asset or not

Usage:

```
ArtifactAsset$is_remote()
```

Method `has_overwrite()`**:** Is the asset will overwrite existing asset with the same name.

Usage:

```
ArtifactAsset$has_overwrite()
```

Method `get_size()`**:** Asset size if the asset is a non-remote asset

Usage:

```
ArtifactAsset$get_size()
```

Method `get_link()`**:** Asset remote link if the asset is remote or NULL

Usage:

```
ArtifactAsset$get_link()
```

Method `get_metadata()`**:** Asset metadata

Usage:

```
ArtifactAsset$get_metadata()
```

Method `get_asset_type()`**:** Asset type

Usage:

```
ArtifactAsset$get_asset_type()
```

`call_api` *Call a Comet REST API endpoint*

Description

This function is only meant for advanced users. If you would like to call any arbitrary Comet API endpoint that isn't natively supported by `cometr`, you can use this function.

Usage

```
call_api(
  endpoint,
  method = c("GET", "POST"),
  params = list(),
  parse_response = TRUE,
  response_json = TRUE,
  local_file_path = NULL,
  api_key = NULL
)
```

Arguments

`endpoint`	The REST API endpoint.
`method`	The HTTP method to use, either "GET" or "POST".
`params`	A list of parameters. For GET endpoints, the parameters are appended to the URL; for POST endpoints, the parameters are sent in the body of the request.
`parse_response`	If `TRUE`, try to parse response from server.
`response_json`	If `TRUE`, try to parse the response as JSON. If `FALSE`, return the response as raw data, e.g. binary.
`local_file_path`	The path to the local file for saving downloaded content if appropriate.
`api_key`	Comet API key (can also be specified using the `COMET_API_KEY` parameter as an environment variable or in a comet config file).

Value

The parsed response

`create_artifact` *Create Comet Artifact object*

Description

Creates new `Artifact` object with provided parameters. After that, the `Artifact` object can be used to save assets and can be logged with an `Experiment`.

Usage

```
create_artifact(
  artifact_name,
  artifact_type,
  artifact_version = NULL,
  aliases = NULL,
  metadata = NULL,
  version_tags = NULL
)
```

Arguments

`artifact_name`	(Required) Artifact name.
`artifact_type`	(Required) The artifact type, for example 'dataset'.
`artifact_version`	The version number to create. If not provided, a new version number will be created automatically.
`aliases`	List of aliases. Some aliases to attach to the future Artifact Version. The aliases list is normalized to remove duplicates.
`metadata`	Some additional meta-data to attach to the future Artifact Version.
`version_tags`	List of tags to be attached to the future Artifact Version.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()

# Create a Comet Artifact
artifact <- create_artifact(artifact_name = "Artifact-Name", artifact_type = "Artifact-Type")
artifact$add("local-file")

exp$log_artifact(artifact)
exp$stop()

## End(Not run)
```

`create_experiment` *Create a new experiment*

Description

Create a new experiment on Comet's servers. The return value is an `Experiment` object that can be used to modify or get information about the experiment. Only one experiment can be active at a time, so make sure to stop an experiment before creating a new one (by calling the `stop()` method on the `Experiment` object).

Usage

```
create_experiment(
  experiment_name = NULL,
  project_name = NULL,
  workspace_name = NULL,
  api_key = NULL,
  keep_active = TRUE,
  log_output = TRUE,
  log_error = FALSE,
  log_code = TRUE,
  log_system_details = TRUE,
  log_git_info = FALSE
)
```

Arguments

`experiment_name`	Experiment name.
`project_name`	Project name (can also be specified using the `COMET_PROJECT_NAME` parameter as an environment variable or in a comet config file).
`workspace_name`	Workspace name (can also be specified using the `COMET_WORKSPACE` parameter as an environment variable or in a comet config file).
`api_key`	Comet API key (can also be specified using the `COMET_API_KEY` parameter as an environment variable or in a comet config file).
`keep_active`	If `TRUE`, automatically send Comet a status update every few seconds until the experiment is stopped to mark the experiment as active on the Comet web dashboard.
`log_output`	If `TRUE`, all standard output will automatically be sent to the Comet servers to display as message logs for the experiment. The output will still be shown in the console as well.
`log_error`	If `TRUE`, all output from 'stderr' (which includes errors, warnings, and messages) will be redirected to the Comet servers to display as message logs for the experiment. Note that unlike `auto_log_output`, if this option is on then these messages will not be shown in the console and instead they will only be logged

to the Comet experiment. This option is set to FALSE by default because of this behavior.

log_code — If TRUE, log the source code of the R script that was called to Comet as the associated code of this experiment. This only works if the you run a script using the Rscript tool and will not work in interactive sessions.

log_system_details — If TRUE, automatically log the system details to Comet when the experiment is created.

log_git_info — If TRUE, log information about the active git repository. Requires the git2r package to be installed.

Value

An Experiment object.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables defined
exp <- create_experiment()
exp$get_key()
exp$get_metadata()
exp$add_tags(c("test", "tag2"))
exp$get_tags()
exp$log_metric("metric1", 5)
exp$get_metric("metric1")
exp$get_metrics_summary()
exp$stop()

## End(Not run)
```

---

create_project *Create a project*

Description

Create a project

Usage

```
create_project(
  project_name,
  project_description,
  public = FALSE,
  workspace_name = NULL,
  api_key = NULL
)
```

Arguments

project_name	Project name.
project_description	Project description.
public	Whether the project should be public or private.
workspace_name	Workspace name (can also be specified using the COMET_WORKSPACE parameter as an environment variable or in a comet config file).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE variables defined
create_project(project_name = "project1", project_description = "My first project")

## End(Not run)
```

---

delete_project *Delete a project*

Description

Delete a project

Usage

```
delete_project(
  project_name,
  delete_experiments = TRUE,
  workspace_name = NULL,
  api_key = NULL
)
```

Arguments

project_name	Project name.
delete_experiments	If TRUE, delete all the experiments in the project.
workspace_name	Workspace name (can also be specified using the COMET_WORKSPACE parameter as an environment variable or in a comet config file).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE variables defined
delete_project(project_name = "project1")

## End(Not run)
```

`disable_logging` *Disable cometr logging*

Description

Generally, if the `COMET_LOGGING_FILE` and `COMET_LOGGING_FILE_LEVEL` parameters are found, then `cometr` will log internal information. You can disable logging for a particular R session by calling `disable_logging()`.

Usage

```
disable_logging()
```

`Experiment` *A Comet Experiment object*

Description

A comet experiment object can be used to modify or get information about an active experiment. All methods documented here are the different ways to interact with an experiment. Use `create_experiment()` to create or `get_experiment()` to retrieve a Comet experiment object.

Value

`LoggedArtifact` with all relevant information about logged artifact.

`LoggedArtifact` with all relevant information about logged artifact.

Methods

Public methods:

- `Experiment$new()`
- `Experiment$get_key()`
- `Experiment$get_workspace_name()`
- `Experiment$get_project_name()`
- `Experiment$get_dynamic()`

- Experiment$get_url()
- Experiment$get_metadata()
- Experiment$archive()
- Experiment$restore()
- Experiment$delete()
- Experiment$stop()
- Experiment$log_metric()
- Experiment$get_metric()
- Experiment$get_metrics_summary()
- Experiment$log_graph()
- Experiment$get_graph()
- Experiment$log_parameter()
- Experiment$get_parameters()
- Experiment$log_other()
- Experiment$get_other()
- Experiment$add_tags()
- Experiment$get_tags()
- Experiment$log_html()
- Experiment$get_html()
- Experiment$upload_asset()
- Experiment$log_remote_asset()
- Experiment$get_asset_list()
- Experiment$get_asset()
- Experiment$create_symlink()
- Experiment$log_git_metadata()
- Experiment$get_git_metadata()
- Experiment$get_git_patch()
- Experiment$get_output()
- Experiment$log_code()
- Experiment$get_code()
- Experiment$log_system_details()
- Experiment$get_system_details()
- Experiment$log_artifact()
- Experiment$get_artifact()
- Experiment$set_start_end_time()
- Experiment$print()

Method `new()`**:** Do not call this function directly. Use `create_experiment()` or `get_experiment()` instead.

Usage:

```
Experiment$new(
  experiment_key,
  experiment_url = NULL,
```

```
  api_key = NULL,
  keep_active = FALSE,
  log_output = FALSE,
  log_error = FALSE,
  dynamic = TRUE,
  workspace_name = NULL,
  project_name = NULL
)
```

Arguments:

`experiment_key` The key of the `Experiment`.

`experiment_url` The URL of the `Experiment`.

`api_key` Comet API key (can also be specified using the `COMET_API_KEY` parameter as an environment variable or in a comet config file).

`keep_active` If `TRUE`, automatically send Comet a status update every few seconds until the experiment is stopped to mark the experiment as active on the Comet web dashboard.

`log_output` If `TRUE`, all standard output will automatically be sent to the Comet servers to display as message logs for the experiment. The output will still be shown in the console as well.

`log_error` If `TRUE`, all output from 'stderr' (which includes errors, warnings, and messages) will be redirected to the Comet servers to display as message logs for the experiment. Note that unlike `auto_log_output`, if this option is on then these messages will not be shown in the console and instead they will only be logged to the Comet experiment. This option is set to `FALSE` by default because of this behavior.

`dynamic` If `TRUE` the Experiment was created rather than retrieved.

`workspace_name` The workspace name (can also be specified using the `COMET_WORKSPACE` parameter as an environment variable or in a comet config file).

`project_name` The project name (can also be specified using the `COMET_PROJECT_NAME` parameter as an environment variable or in a comet config file).

Method `get_key()`**:** Get the experiment key of an experiment.

Usage:

```
Experiment$get_key()
```

Method `get_workspace_name()`**:** Get the workspace name of an experiment.

Usage:

```
Experiment$get_workspace_name()
```

Method `get_project_name()`**:** Get the project name of an experiment.

Usage:

```
Experiment$get_project_name()
```

Method `get_dynamic()`**:** Get the dynamic status of an experiment.

Usage:

```
Experiment$get_dynamic()
```

Method `get_url()`**:** Get the URL to view an experiment in the browser.

Usage:

```
Experiment$get_url()
```

Method `get_metadata()`**:** Get an experiment's metadata.

Usage:

```
Experiment$get_metadata()
```

Method `archive()`**:** Archive an experiment.

Usage:

```
Experiment$archive()
```

Method `restore()`**:** Restore an archived experiment.

Usage:

```
Experiment$restore()
```

Method `delete()`**:** Delete an experiment.

Usage:

```
Experiment$delete()
```

Method `stop()`**:** Stop an experiment. Always call this method before creating a new experiment.

Usage:

```
Experiment$stop()
```

Method `log_metric()`**:** Log a metric name and value. Metrics are the only items that are logged as a full time series. However, even metrics can be throttled if too much data (either by rate or by count) is attempted to be logged.

Usage:

```
Experiment$log_metric(name, value, step = NULL, epoch = NULL, context = NULL)
```

Arguments:

`name` (Required) Name of the metric.

`value` (Required) Value of the metric.

`step` Step number.

`epoch` Epoch.

`context` Context.

Method `get_metric()`**:** Get All Metrics For Name

Usage:

```
Experiment$get_metric(name)
```

Arguments:

`name` (Required) Name of metric.

Method `get_metrics_summary()`**:** Get an experiment's metrics summary.

Usage:

```
Experiment$get_metrics_summary()
```

Method `log_graph()`**:** Log an experiment's associated model graph.

Usage:

```
Experiment$log_graph(graph)
```

Arguments:

`graph` (Required) JSON representation of a graph.

Method `get_graph()`**:** Get an experiment's model graph.

Usage:

```
Experiment$get_graph()
```

Method `log_parameter()`**:** Log a parameter name and value. Note that you can only retrieve parameters summary data (e.g., this is not recorded as a full time series).

Usage:

```
Experiment$log_parameter(name, value, step = NULL)
```

Arguments:

`name` (Required) Name of the parameter.

`value` (Required) Value of the parameter.

`step` Step number.

Method `get_parameters()`**:** Get an experiment's parameters summary.

Usage:

```
Experiment$get_parameters()
```

Method `log_other()`**:** Log a key/value 'other" data (not a metric or parameter). Note that you can only retrieve others summary data (e.g., this is not recorded as a full time series).

Usage:

```
Experiment$log_other(key, value)
```

Arguments:

`key` (Required) The key.

`value` (Required) The value.

Method `get_other()`**:** Get an experiment's others (logged with `log_other()`) summary.

Usage:

```
Experiment$get_other()
```

Method `add_tags()`**:** Add a list of tags to an experiment.

Usage:

```
Experiment$add_tags(tags)
```

Arguments:

`tags` (Required) List of tags.

Method `get_tags()`**:** Get an experiment's tags.

Usage:

```
Experiment$get_tags()
```

Method `log_html()`**:** Set (or append onto) an experiment's HTML.

Usage:

```
Experiment$log_html(html, override = FALSE)
```

Arguments:

`html` (Required) An HTML string to add to the experiment.

`override` If `TRUE`, override the previous HTML. If `FALSE`, append to it.

Method `get_html()`**:** Get an experiment's HTML.

Usage:

```
Experiment$get_html()
```

Method `upload_asset()`**:** Upload a file to the experiment.

Usage:

```
Experiment$upload_asset(
  file,
  step = NULL,
  overwrite = NULL,
  context = NULL,
  type = NULL,
  name = NULL,
  metadata = NULL
)
```

Arguments:

`file` (Required) Path to the file to upload.

`step` Step number.

`overwrite` If `TRUE`, overwrite any uploaded file with the same name.

`context` The context.

`type` The type of asset.

`name` Name of the file on comet. By default the name of the file will match the file that you upload, but you can use this parameter to use a different name.

`metadata` Metadata to upload along with the file.

Method `log_remote_asset()`**:** Logs a Remote Asset identified by an URI. A Remote Asset is an asset but its content is not uploaded and stored on Comet. Rather a link for its location is stored, so you can identify and distinguish between two experiment using different version of a dataset stored somewhere else.

Usage:

```
Experiment$log_remote_asset(
  uri,
  remote_file_name = NULL,
  step = NULL,
  overwrite = FALSE,
  type = "asset",
  metadata = NULL
)
```

Arguments:

`uri` (Required) The remote asset location, there is no imposed format, and it could be a private link.

`remote_file_name` The "name" of the remote asset, could be a dataset name, a model file name.

`step` Step number.

`overwrite` If `TRUE`, overwrite any logged asset with the same name.

`type` The type of asset, default: "asset".

`metadata` Metadata to log along with the asset

Method `get_asset_list()`**:** Get an experiment's asset list.

Usage:

```
Experiment$get_asset_list(type = NULL)
```

Arguments:

`type` The type of assets to retrieve (by default, all assets are returned).

Method `get_asset()`**:** Get an asset.

Usage:

```
Experiment$get_asset(assetId)
```

Arguments:

`assetId` (Required) The asset ID to retrieve.

Method `create_symlink()`**:** Add a symlink to an experiment in another project.

Usage:

```
Experiment$create_symlink(project_name)
```

Arguments:

`project_name` (Required) Project that the experiment to should linked to.

Method `log_git_metadata()`**:** Log an experiment's git metadata. This should only be called once and it can be done automatically by enabling `log_git_info` in `create_experiment()` or `get_experiment()`. This will replace any previous git metadata that was logged.

Usage:

```
Experiment$log_git_metadata(
  branch = NULL,
  origin = NULL,
  parent = NULL,
  user = NULL,
  root = NULL
)
```

Arguments:

`branch` Git branch name.

`origin` Git repository origin.

`parent` Git commit SHA.

`user` Git username.

root Git root.

Method `get_git_metadata()`**:** Get the git metadata of an experiment.

Usage:

```
Experiment$get_git_metadata()
```

Method `get_git_patch()`**:** Get the git patch of an experiment.

Usage:

```
Experiment$get_git_patch()
```

Method `get_output()`**:** Get an experiment's standard output and error.

Usage:

```
Experiment$get_output()
```

Method `log_code()`**:** Log an experiment's source code. This should only be called once and it can be done automatically by enabling `log_code` in `create_experiment()` or `get_experiment()`. This will replace any previous code that was logged.

Usage:

```
Experiment$log_code(code)
```

Arguments:

code The code to set as the source code.

Method `get_code()`**:** Get an experiment's source code.

Usage:

```
Experiment$get_code()
```

Method `log_system_details()`**:** Log system details. This can be done automatically by enabling `log_system_details` in `create_experiment()` or `get_experiment()`.

Usage:

```
Experiment$log_system_details(
  command = NULL,
  executable = NULL,
  hostname = NULL,
  installed_packages = NULL,
  gpu_static_info = NULL,
  ip = NULL,
  network_interface_ips = NULL,
  additional_system_info = NULL,
  os = NULL,
  os_packages = NULL,
  os_type = NULL,
  pid = NULL,
  user = NULL,
  r_version = NULL,
  r_version_verbose = NULL
)
```

Arguments:

`command` Script and optional arguments.

`executable` Executable.

`hostname` Hostname.

`installed_packages` List of installed R packages.

`gpu_static_info` List of GPU information, where each GPU is a `list()` with fields `gpuIndex`, `name`, `powerLimit`, `totalMemory`, `uuid`.

`ip` IP address.

`network_interface_ips` List of network interface IPs.

`additional_system_info` List of additional parameters to log, where each parameter is a `list()` with `key` and `value` pairs.

`os` Full details about operating system.

`os_packages` List of operating system packages installed.

`os_type` Operating system type.

`pid` Process ID.

`user` User.

`r_version` Short form R version.

`r_version_verbose` Long form R version.

Method `get_system_details()`**:** Get an experiment's system details.

Usage:

```
Experiment$get_system_details()
```

Method `log_artifact()`**:** Log an `Artifact` object, synchronously create a new Artifact Version and upload all local and remote assets attached to the `Artifact` object.

Usage:

```
Experiment$log_artifact(artifact)
```

Arguments:

`artifact` an `Artifact` object.

Method `get_artifact()`**:** Returns a logged artifact object that can be used to access the artifact version assets and download them locally.

If no version or alias is provided, the latest version for that artifact is returned.

Usage:

```
Experiment$get_artifact(
  artifact_name,
  workspace = NULL,
  version_or_alias = NULL
)
```

Arguments:

`artifact_name` (Required) Retrieve an artifact with that name. This could either be a fully qualified artifact name like `workspace/artifact-name:versionOrAlias` or just the name of the artifact like `artifact-name`.

`workspace` Retrieve an artifact belonging to that workspace.

`version_or_alias` Retrieve the artifact by the given alias or version.

Examples:

```
\dontrun{
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()

# Get a Comet Artifact
logged_artifact <- exp$get_artifact("workspace/artifact-name:version_or_alias")

# Which is equivalent to
logged_artifact = exp$get_artifact(artifact_name="artifact-name",
                                   workspace="workspace",
                                   version_or_alias="version_or_alias")
}
```

Method `set_start_end_time()`**:** Set an experiment's start and end time.

Usage:

```
Experiment$set_start_end_time(start = NULL, end = NULL)
```

Arguments:

`start` Start time for the experiment (milliseconds since the Epoch)

`end` End time for the experiment (milliseconds since the Epoch)

Method `print()`**:** Print the experiment.

Usage:

```
Experiment$print()
```

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()
exp$get_key()
exp$get_metadata()
exp$add_tags(c("test", "tag2"))
exp$get_tags()
exp$log_metric("metric1", 5)
exp$get_metric("metric1")
exp$get_metrics_summary()
exp$stop()

## End(Not run)


## ------------------------------------------------
## Method `Experiment$get_artifact`
## ------------------------------------------------
```

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()

# Get a Comet Artifact
logged_artifact <- exp$get_artifact("workspace/artifact-name:version_or_alias")

# Which is equivalent to
logged_artifact = exp$get_artifact(artifact_name="artifact-name",
                                   workspace="workspace",
                                   version_or_alias="version_or_alias")

## End(Not run)
```

`get_api_version` *Get the Comet API version*

Description

Get the Comet API version

Usage

```
get_api_version()
```

`get_columns` *Get a project's columns*

Description

Either `project_id` should be provided, or both `project_name` and `workspace_name` should be provided. If `project_id` is provided, then `project_name` and `workspace_name` are ignored.

Usage

```
get_columns(
  project_id = NULL,
  project_name = NULL,
  workspace_name = NULL,
  api_key = NULL,
  archived = FALSE
)
```

Arguments

project_id	Project ID.
project_name	Project name (can also be specified using the COMET_PROJECT_NAME parameter as an environment variable or in a comet config file).
workspace_name	Workspace name (can also be specified using the COMET_WORKSPACE parameter as an environment variable or in a comet config file).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).
archived	If TRUE, retrieve archived experiments. Otherwise, retrieve active experiments.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables defined
get_columns()

## End(Not run)
```

---

get_experiment *Get a previously created experiment*

Description

Get a previously created experiment on Comet's servers. The return value is an Experiment object that can be used to modify or get information about the experiment.

Usage

```
get_experiment(
  experiment_key,
  api_key = NULL,
  keep_active = FALSE,
  log_output = FALSE,
  log_error = FALSE,
  log_code = FALSE,
  log_system_details = FALSE,
  log_git_info = FALSE
)
```

Arguments

experiment_key	Experiment key.
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).

keep_active — if TRUE keeps a communication channel open with comet.ml

log_output — If TRUE, all standard output will automatically be sent to the Comet servers to display as message logs for the experiment. The output will still be shown in the console as well.

log_error — If TRUE, all output from 'stderr' (which includes errors, warnings, and messages) will be redirected to the Comet servers to display as message logs for the experiment. Note that unlike auto_log_output, if this option is on then these messages will not be shown in the console and instead they will only be logged to the Comet experiment. This option is set to FALSE by default because of this behavior.

log_code — If TRUE, log the source code of the R script that was called to Comet as the associated code of this experiment. This only works if the you run a script using the Rscript tool and will not work in interactive sessions.

log_system_details — If TRUE, automatically log the system details to Comet when the experiment is created.

log_git_info — If TRUE, log information about the active git repository. Requires the git2r package to be installed.

Value

An Experiment object.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables defined
exp <- get_experiment("SOME-EXPERIMENT-KEY")
exp$get_key()
exp$get_metadata()
exp$add_tags(c("test", "tag2"))
exp$get_tags()
exp$log_metric("metric1", 5)
exp$get_metric("metric1")
exp$get_metrics_summary()
exp$stop()

## End(Not run)
```

---

get_experiments *Get a project's experiments*

Description

Either project_id should be provided, or both project_name and workspace_name should be provided. If project_id is provided, then project_name and workspace_name are ignored.

Usage

```
get_experiments(
  project_id = NULL,
  project_name = NULL,
  workspace_name = NULL,
  api_key = NULL,
  archived = FALSE
)
```

Arguments

project_id	Project ID.
project_name	Project name (can also be specified using the COMET_PROJECT_NAME parameter as an environment variable or in a comet config file).
workspace_name	Workspace name (can also be specified using the COMET_WORKSPACE parameter as an environment variable or in a comet config file).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).
archived	If TRUE, retrieve archived experiments. Otherwise, retrieve active experiments.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables defined
get_experiments()

## End(Not run)
```

---

get_multi_metric_chart

Get Multi-Metric Chart

Description

Get Multi-Metric Chart

Usage

```
get_multi_metric_chart(
  experiment_keys,
  metrics = list(),
  params = list(),
  full = TRUE,
  independent = TRUE,
  api_key = NULL
)
```

Arguments

experiment_keys	List of experiment keys.
metrics	List of metric names to retrieve.
params	List of parameter names to retrieve.
full	Whether to fetch all values (up to 15,000) or a sampled subset (about 500 points).
independent	Whether the metrics should be fetched individually or as a correlated whole (only return values for steps for which you have values for every requested metric name).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY variable defined
experiment <- "<your experiment key>"
metrics <- c("<metric1>", "<metric2>")
get_multi_metric_chart(experiment_keys = experiment, metrics = metrics)

## End(Not run)
```

---

get_projects *Get a workspace's projects*

Description

Get a workspace's projects

Usage

```
get_projects(workspace_name = NULL, api_key = NULL)
```

Arguments

workspace_name	Workspace name (can also be specified using the COMET_WORKSPACE parameter as an environment variable or in a comet config file).
api_key	Comet API key (can also be specified using the COMET_API_KEY parameter as an environment variable or in a comet config file).

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE variables defined
get_projects()

## End(Not run)
```

`get_workspaces` *Get a user's workspaces*

Description

Get a user's workspaces

Usage

```
get_workspaces(api_key = NULL)
```

Arguments

`api_key`	Comet API key (can also be specified using the `COMET_API_KEY` parameter as an environment variable or in a comet config file).

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY variable defined
get_workspaces()

## End(Not run)
```

`LoggedArtifact` *A Logged Comet Artifact object*

Description

Comet Artifacts allow keeping track of assets beyond any particular experiment. The `LoggedArtifact` is a Comet `Artifact` that already logged to the Comet servers and can be used to access the artifact version assets and download them locally.

Methods

Public methods:

- LoggedArtifact$new()
- LoggedArtifact$get_artifact_name()
- LoggedArtifact$get_artifact_type()
- LoggedArtifact$get_artifact_version()
- LoggedArtifact$get_artifact_id()
- LoggedArtifact$get_artifact_tags()
- LoggedArtifact$get_aliases()
- LoggedArtifact$get_metadata()
- LoggedArtifact$get_version_tags()
- LoggedArtifact$get_workspace()
- LoggedArtifact$get_artifact_version_id()
- LoggedArtifact$get_source_experiment_key()
- LoggedArtifact$get_experiment_key()
- LoggedArtifact$size()
- LoggedArtifact$get_assets()
- LoggedArtifact$get_remote_assets()
- LoggedArtifact$update_artifact_tags()
- LoggedArtifact$update_version_tags()
- LoggedArtifact$update_aliases()
- LoggedArtifact$download()

Method `new()`**:** Creates new `LoggedArtifact` object with provided parameters. Do not use this method directly. Use `Experiment$get_artifact()` to retrieve `LoggedArtifact`.

Usage:

```
LoggedArtifact$new(
  artifact_name,
  artifact_type,
  artifact_id,
  artifact_version_id,
  workspace,
  experiment_key,
  artifact_version,
  aliases,
  artifact_tags,
  version_tags,
  size,
  metadata = NULL,
  source_experiment_key = NULL
)
```

Arguments:

`artifact_name` (Required) Artifact name.

`artifact_type` (Required) The artifact type.

`artifact_id` (Required) The ID of artifact.

`artifact_version_id` (Required) The ID of Artifact Version.

`workspace` (Required) The workspace where artifact saved.

`experiment_key` (Required) The ID of the associated experiment.

`artifact_version` (Required) The latest artifact version.

`aliases` (Required) List of Artifact Version aliases.

`artifact_tags` (Required) The list of artifact tags.

`version_tags` (Required) List of Artifact Version tags.

`size` (Required) The total size of logged artifact version. It is the sum of all the artifact version assets.

`metadata` The meta-data of Artifact Version.

`source_experiment_key` The ID of the experiment that created this artifact version.

Method `get_artifact_name()`**:** Get the name of the artifact.

Usage:

```
LoggedArtifact$get_artifact_name()
```

Method `get_artifact_type()`**:** Get the type of the artifact.

Usage:

```
LoggedArtifact$get_artifact_type()
```

Method `get_artifact_version()`**:** Get the version of the artifact.

Usage:

```
LoggedArtifact$get_artifact_version()
```

Method `get_artifact_id()`**:** Get the ID of the artifact.

Usage:

```
LoggedArtifact$get_artifact_id()
```

Method `get_artifact_tags()`**:** Get the tags of the artifact.

Usage:

```
LoggedArtifact$get_artifact_tags()
```

Method `get_aliases()`**:** Get the version of the artifact.

Usage:

```
LoggedArtifact$get_aliases()
```

Method `get_metadata()`**:** Get the metadata of the artifact.

Usage:

```
LoggedArtifact$get_metadata()
```

Method `get_version_tags()`**:** Get the list of tags of the artifact version.

Usage:

```
LoggedArtifact$get_version_tags()
```

Method `get_workspace()`: Get the workspace of the Artifact.

Usage:

```
LoggedArtifact$get_workspace()
```

Method `get_artifact_version_id()`: The ID of current Artifact Version

Usage:

```
LoggedArtifact$get_artifact_version_id()
```

Method `get_source_experiment_key()`: The ID of the experiment that created this artifact version.

Usage:

```
LoggedArtifact$get_source_experiment_key()
```

Method `get_experiment_key()`: The ID of the associated experiment.

Usage:

```
LoggedArtifact$get_experiment_key()
```

Method `size()`: Get/set artifact size.

Usage:

```
LoggedArtifact$size(size = NULL)
```

Arguments:

`size` The new size for the Artifact or `NULL` if retrieving existing size of the Artifact.

Method `get_assets()`: Get the list of all `LoggedArtifactAsset` that have been logged with this `LoggedArtifact` from Comet server.

Usage:

```
LoggedArtifact$get_assets()
```

Method `get_remote_assets()`: Get the list of remote `LoggedArtifactAsset` that have been logged with this `LoggedArtifact` from Comet server.

Usage:

```
LoggedArtifact$get_remote_assets()
```

Method `update_artifact_tags()`: Update the logged artifact tags

Usage:

```
LoggedArtifact$update_artifact_tags(artifact_tags)
```

Arguments:

`artifact_tags` The new tags for the artifact

Method `update_version_tags()`: Update the logged artifact version tags

Usage:

```
LoggedArtifact$update_version_tags(version_tags)
```

Arguments:

`version_tags` The new tags for the artifact version

Method `update_aliases()`**:** Update the logged artifact version aliases

Usage:

```
LoggedArtifact$update_aliases(aliases)
```

Arguments:

`aliases` The new aliases for the artifact version

Method `download()`**:** Download the current Artifact Version assets to a given directory (or the local directory by default). This downloads only non-remote assets.

Usage:

```
LoggedArtifact$download(path = NULL, overwrite_strategy = FALSE)
```

Arguments:

`path` Where to download artifact version assets. If not provided, a temporary path will be used.

`overwrite_strategy` One of the three possible strategies to handle conflict when trying to download an artifact version asset to a path with an existing file. See below for allowed values. Default is False or "FAIL".
Overwrite strategy allowed values:

- False or "FAIL": If a file already exists and its content is different, raise the `comet_ml.exceptions.ArtifactDown`
- "PRESERVE": If a file already exists and its content is different, show a WARNING but preserve the existing content.
- True or "OVERWRITE": If a file already exists and its content is different, replace it by the asset version asset.

Returns: `Artifact` object.

Examples

```
## Not run:
library(cometr)
# Assuming you have COMET_API_KEY, COMET_WORKSPACE, COMET_PROJECT_NAME variables define
exp <- create_experiment()

# Get a Comet Artifact
artifact <- exp$get_artifact(artifact_name = "workspace/artifact-name:versionOrAlias")

exp$stop()

## End(Not run)
```

`LoggedArtifactAsset` *An Artifact Asset object that was already logged.*

Description

The `LoggedArtifactAsset` represent local or remote asset already logged with particular `Artifact` to the Comet.

Super class

`cometr::ArtifactAsset` -> `LoggedArtifactAsset`

Methods

Public methods:

- `LoggedArtifactAsset$new()`
- `LoggedArtifactAsset$get_id()`
- `LoggedArtifactAsset$get_artifact_version_id()`
- `LoggedArtifactAsset$get_artifact_id()`
- `LoggedArtifactAsset$download()`

Method `new()`**:** Creates a new `LoggedArtifactAsset` object with provided parameters.

Usage:

```
LoggedArtifactAsset$new(
  logical_path,
  remote,
  size,
  metadata,
  asset_type,
  id,
  artifact_version_id,
  artifact_id,
  experiment_key,
  link = NULL
)
```

Arguments:

`logical_path` the logical file name.

`remote` Is the asset a remote asset or not.

`size` The size if the asset of a non-remote asset.

`metadata` The metadata to be associated with the asset.

`asset_type` The type of asset.

`id` The ID of the asset

`artifact_version_id` The ID of Artifact Version associated with this asset.

`artifact_id` The ID of Artifact associated with this asset.

`experiment_key` The experiment key of the experiment that logged this asset.

`link` The remote link if the asset is remote.

Method `get_id()`**:** Asset unique ID

Usage:

```
LoggedArtifactAsset$get_id()
```

Method `get_artifact_version_id()`**:** The ID of Artifact Version associated with this asset

Usage:

```
LoggedArtifactAsset$get_artifact_version_id()
```

Method `get_artifact_id()`**:** The ID of Artifact associated with this asset

Usage:

```
LoggedArtifactAsset$get_artifact_id()
```

Method `download()`**:** Download the asset to a given full path or directory.

Usage:

```
LoggedArtifactAsset$download(
  local_path = NULL,
  logical_path = NULL,
  overwrite_strategy = FALSE
)
```

Arguments:

`local_path` The root folder to which to download. If `NULL`, will download to a tmp path, otherwise will be either a root local path or a full local path.

`logical_path` The path relative to the root local_path to use If `NULL` and `local_path==NULL` then no relative path is used, file would just be a tmp path on local disk. If `NULL` and `local_path!=NULL` then the local_path will be treated as a root path, and the asset's `logical_path` will be appended to the root path to form a full local path.

`overwrite_strategy` can be `FALSE`, "FAIL", "PRESERVE" or "OVERWRITE" and follows the same semantics for overwrite strategy as artifact.download()

Returns: `ArtifactAsset` holding information about downloaded asset data file.

Index

Artifact, 2, *5*, *20*, *27*, *31*
ArtifactAsset, *4*, 5

call_api, 7
create_artifact, 8
create_experiment, 9
create_experiment(), *12*, *18, 19*
create_project, 10

delete_project, 11
disable_logging, 12

Experiment, *2*, *8–10*, 12, *23, 24*

get_api_version, 22
get_columns, 22
get_experiment, 23
get_experiment(), *12*, *18, 19*
get_experiments, 24
get_multi_metric_chart, 25
get_projects, 26
get_workspaces, 27

LoggedArtifact, *12*, 27
LoggedArtifactAsset, *30*, 31

TRUE, *4*, *6*