

Package ‘cometr’

October 12, 2022

Title 'Comet' API for R

Version 0.2.0

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.ml>.

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

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

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Dean Attali [aut],
Doug Blank [cre]

Maintainer Doug Blank <doug@comet.ml>

Repository CRAN

Date/Publication 2020-08-13 17:40:03 UTC

R topics documented:

call_api	2
create_experiment	3
create_project	4
delete_project	5
disable_logging	6
Experiment	6

get_api_version	13
get_columns	14
get_experiment	15
get_experiments	16
get_multi_metric_chart	17
get_projects	18
get_workspaces	19

Index	20
--------------	-----------

call_api	<i>Call a Comet REST API endpoint</i>
----------	---------------------------------------

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(),
  response_json = TRUE,
  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.
response_json	If TRUE, try to parse the response as JSON. If FALSE, return the response as raw data (useful when the response is a 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).

Value

The parsed response

create_experiment	<i>Create a new experiment</i>
-------------------	--------------------------------

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 <code>auto_log_output</code> , 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 behaviour.
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	<i>Create a project</i>
----------------	-------------------------

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	<i>Delete a project</i>
----------------	-------------------------

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	<i>Disable cometr logging</i>
-----------------	-------------------------------

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	<i>A Comet Experiment object</i>
------------	----------------------------------

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.

Methods**Public methods:**

- [Experiment\\$new\(\)](#)
- [Experiment\\$get_key\(\)](#)
- [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$upload_asset()`
- `Experiment$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$set_start_end_time()`
- `Experiment$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  
)
```

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

Usage:

```
Experiment$get_key()
```

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 `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 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)
```

get_api_version

Get the Comet API version

Description

Get the Comet API version

Usage

```
get_api_version()
```

get_columns	<i>Get a project's columns</i>
-------------	--------------------------------

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

<code>project_id</code>	Project ID.
<code>project_name</code>	Project name (can also be specified using the <code>COMET_PROJECT_NAME</code> parameter as an environment variable or in a comet config file).
<code>workspace_name</code>	Workspace name (can also be specified using the <code>COMET_WORKSPACE</code> parameter as an environment variable or in a comet config file).
<code>api_key</code>	Comet API key (can also be specified using the <code>COMET_API_KEY</code> parameter as an environment variable or in a comet config file).
<code>archived</code>	If <code>TRUE</code> , 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	<i>get_experiment</i>
----------------	-----------------------

Description

Get a previously created experiment

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 behaviour.
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

<code>project_id</code>	Project ID.
<code>project_name</code>	Project name (can also be specified using the <code>COMET_PROJECT_NAME</code> parameter as an environment variable or in a comet config file).
<code>workspace_name</code>	Workspace name (can also be specified using the <code>COMET_WORKSPACE</code> parameter as an environment variable or in a comet config file).
<code>api_key</code>	Comet API key (can also be specified using the <code>COMET_API_KEY</code> parameter as an environment variable or in a comet config file).
<code>archived</code>	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	<i>Get a user's workspaces</i>
----------------	--------------------------------

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)
```

Index

call_api, [2](#)
create_experiment, [3](#)
create_experiment(), [6](#), [11](#), [12](#)
create_project, [4](#)

delete_project, [5](#)
disable_logging, [6](#)

Experiment, [3](#), [4](#), [6](#), [15](#), [16](#)

get_api_version, [13](#)
get_columns, [14](#)
get_experiment, [15](#)
get_experiment(), [6](#), [11](#), [12](#)
get_experiments, [16](#)
get_multi_metric_chart, [17](#)
get_projects, [18](#)
get_workspaces, [19](#)