

# Package ‘clustermq’

January 10, 2025

**Title** Evaluate Function Calls on HPC Schedulers (LSF, SGE, SLURM, PBS/Torque)

**Version** 0.9.6

**Maintainer** Michael Schubert <mschu.dev@gmail.com>

**Description** Evaluate arbitrary function calls using workers on HPC schedulers in single line of code. All processing is done on the network without accessing the file system. Remote schedulers are supported via SSH.

**URL** <https://mschubert.github.io/clustermq/>

**BugReports** <https://github.com/mschubert/clustermq/issues>

**SystemRequirements** ZeroMQ (libzmq) >= 4.3.0 (optional; otherwise bundled)

**Depends** R (>= 3.6.2)

**LinkingTo** Rcpp

**Imports** methods, narray, globals, progress, R6, Rcpp, utils

**License** Apache License (== 2.0) | file LICENSE

**Encoding** UTF-8

**Suggests** BiocParallel, callr, devtools, foreach, iterators, knitr, parallel, rmarkdown, roxygen2 (>= 5.0.0), testthat, tools

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**NeedsCompilation** yes

**Author** Michael Schubert [aut, cre, cph]  
(<<https://orcid.org/0000-0002-6862-5221>>),  
ZeroMQ authors [aut, cph] (source files in 'src/libzmq' and 'src/cppzmq')

**Repository** CRAN

**Date/Publication** 2025-01-10 16:30:06 UTC

## Contents

Q	2
Q_rows	3
register_dopar_cmq	5
workers	5
<b>Index</b>	<b>7</b>

---

### Q *Queue function calls on the cluster*

---

#### Description

Queue function calls on the cluster

#### Usage

```
Q(
  fun,
  ...,
  const = list(),
  export = list(),
  pkgs = c(),
  seed = 128965,
  memory = NULL,
  template = list(),
  n_jobs = NULL,
  job_size = NULL,
  split_array_by = -1,
  rettype = "list",
  fail_on_error = TRUE,
  workers = NULL,
  log_worker = FALSE,
  chunk_size = NA,
  timeout = Inf,
  max_calls_worker = Inf,
  verbose = TRUE
)
```

#### Arguments

fun	A function to call
...	Objects to be iterated in each function call
const	A list of constant arguments passed to each function call
export	List of objects to be exported to the worker
pkgs	Character vector of packages to load on the worker

seed	A seed to set for each function call
memory	Short for 'template=list(memory=value)'
template	A named list of values to fill in the scheduler template
n_jobs	The number of jobs to submit; upper limit of jobs if job_size is given as well
job_size	The number of function calls per job
split_array_by	The dimension number to split any arrays in '...'; default: last
rettype	Return type of function call (vector type or 'list')
fail_on_error	If an error occurs on the workers, continue or fail?
workers	Optional instance of QSys representing a worker pool
log_worker	Write a log file for each worker
chunk_size	Number of function calls to chunk together defaults to 100 chunks per worker or max. 10 kb per chunk
timeout	Maximum time in seconds to wait for worker (default: Inf)
max_calls_worker	Maximum number of chunks that will be sent to one worker
verbose	Print status messages and progress bar (default: TRUE)

**Value**

A list of whatever 'fun' returned

**Examples**

```
## Not run:
# Run a simple multiplication for numbers 1 to 3 on a worker node
fx = function(x) x * 2
Q(fx, x=1:3, n_jobs=1)
# list(2,4,6)

# Run a mutate() call in dplyr on a worker node
iris %>%
  mutate(area = Q(`*`, e1=Sepal.Length, e2=Sepal.Width, n_jobs=1))
# iris with an additional column 'area'

## End(Not run)
```

---

Q\_rows

*Queue function calls defined by rows in a data.frame*

---

**Description**

Queue function calls defined by rows in a data.frame

**Usage**

```

Q_rows(
  df,
  fun,
  const = list(),
  export = list(),
  pkgs = c(),
  seed = 128965,
  memory = NULL,
  template = list(),
  n_jobs = NULL,
  job_size = NULL,
  rettype = "list",
  fail_on_error = TRUE,
  workers = NULL,
  log_worker = FALSE,
  chunk_size = NA,
  timeout = Inf,
  max_calls_worker = Inf,
  verbose = TRUE
)

```

**Arguments**

df	data.frame with iterated arguments
fun	A function to call
const	A list of constant arguments passed to each function call
export	List of objects to be exported to the worker
pkgs	Character vector of packages to load on the worker
seed	A seed to set for each function call
memory	Short for ‘template=list(memory=value)’
template	A named list of values to fill in the scheduler template
n_jobs	The number of jobs to submit; upper limit of jobs if job_size is given as well
job_size	The number of function calls per job
rettype	Return type of function call (vector type or ‘list’)
fail_on_error	If an error occurs on the workers, continue or fail?
workers	Optional instance of QSys representing a worker pool
log_worker	Write a log file for each worker
chunk_size	Number of function calls to chunk together defaults to 100 chunks per worker or max. 10 kb per chunk
timeout	Maximum time in seconds to wait for worker (default: Inf)
max_calls_worker	Maximum number of chunks that will be sent to one worker
verbose	Print status messages and progress bar (default: TRUE)

**Examples**

```
## Not run:
# Run a simple multiplication for data frame columns x and y on a worker node
fx = function (x, y) x * y
df = data.frame(x = 5, y = 10)
Q_rows(df, fx, job_size = 1)
# [1] 50

# Q_rows also matches the names of a data frame with the function arguments
fx = function (x, y) x - y
df = data.frame(y = 5, x = 10)
Q_rows(df, fx, job_size = 1)
# [1] 5

## End(Not run)
```

---

register_dopar_cmq	<i>Register clustermq as 'foreach' parallel handler</i>
--------------------	---------------------------------------------------------

---

**Description**

Register clustermq as 'foreach' parallel handler

**Usage**

```
register_dopar_cmq(...)
```

**Arguments**

... List of arguments passed to the 'Q' function, e.g. n\_jobs

---

workers	<i>Creates a pool of workers</i>
---------	----------------------------------

---

**Description**

Creates a pool of workers

**Usage**

```
workers(
  n_jobs,
  data = NULL,
  reuse = TRUE,
  template = list(),
  log_worker = FALSE,
```

```
    qsys_id = getOption("clustermq.scheduler", qsys_default),  
    verbose = FALSE,  
    ...  
  )
```

**Arguments**

n_jobs	Number of jobs to submit (0 implies local processing)
data	Set common data (function, constant args, seed)
reuse	Whether workers are reusable or get shut down after call
template	A named list of values to fill in template
log_worker	Write a log file for each worker
qsys_id	Character string of QSys class to use
verbose	Print message about worker startup
...	Additional arguments passed to the qsys constructor

**Value**

An instance of the QSys class

# Index

Q, [2](#)

Q\_rows, [3](#)

register\_dopar\_cmq, [5](#)

workers, [5](#)