Package ‘bsub’

October 12, 2022

Type Package
Title Submitter and Monitor of the 'LSF Cluster'
Version 1.1.0
Date 2021-06-30
Author Zuguang Gu
Maintainer Zuguang Gu <z.gu@dkfz.de>
Depends R (>= 3.0.0)
Imports GlobalOptions (>= 0.1.1), GetoptLong (>= 0.1.8), digest,
       utils, stats, clisymbols, crayon, methods, grDevices, graphics
Suggests knitr, ssh, DT (>= 0.13), shiny (>= 1.0.0), shinyjqui,
       igraph, graph, Rgraphviz, ggplot2, rmarkdown, markdown
biocViews Software, Infrastructure
Description It submits R code/R scripts/shell commands to 'LSF cluster'
       (<https://en.wikipedia.org/wiki/Platform_LSF>, the 'bsub' system) without
       leaving R. There is also an interactive 'shiny' app for monitoring the job status.
VignetteBuilder knitr
URL https://github.com/jokergoo/bsub
License MIT + file LICENSE
NeedsCompilation no
SystemRequirements Platform LSF, bsub
Repository CRAN
Date/Publication 2021-07-01 15:50:10 UTC

R topics documented:

   bconf .......................................................... 2
   bjobs .......................................................... 3
   bjobs_barplot .............................................. 4
   bjobs_done ................................................ 5
   bjobs_exit .................................................. 5
### bconf

Print current configuration

#### Description

Print current configuration

#### Usage

```r
bconf
```

#### Details

This function is only for printing. Use `bsub_opt` to change configurations.

You simply type `bconf` (without the brackets) in the interactive R console.

#### Value

A `bconf` object.
Examples

bconf

---

bjobs

Summary of jobs

Description

Summary of jobs

Usage

```r
bjobs(status = c("RUN", "PEND"), max = Inf, filter = NULL, print = TRUE)
```

Arguments

- **status**: Status of the jobs. Use "all" for all jobs.
- **max**: Maximal number of recent jobs.
- **filter**: Regular expression to filter on job names.
- **print**: Whether to print the table.

Details

There is an additional column "RECENT" which is the order for the job with the same name. 1 means the most recent job.

You can directly type `bjobs` without parentheses which runs `bjobs` with defaults.

Value

A data frame with selected job summaries.

See Also

- `brecent` shows the most recent.
- `bjobs_done` shows the "DONE" jobs.
- `bjobs_exit` shows the "EXIT" jobs.
- `bjobs_pending` shows the "PEND" jobs.
- `bjobs_running` shows the "RUN" jobs.
Examples

```r
## Not run:
bjobs # this is the same as bjobs()
bjobs() # all running and pending jobs
bjobs(status = "all") # all jobs
bjobs(status = "RUN") # all running jobs, you can also use 'bjobs_running'
bjobs(status = "PEND") # all pending jobs, you can also use 'bjobs_pending'
bjobs(status = "DONE") # all done jobs, you can also use 'bjobs_done'
bjobs(status = "EXIT") # all exit jobs, you can also use 'bjobs_exit'
bjobs(status = "all", max = 20) # last 20 jobs
bjobs(status = "DONE", filter = "example") # done jobs with name '.*example.*'

## End(Not run)
```

---

**bjobs_barplot**  
**Barplot of number of jobs**

### Description

Barplot of number of jobs

### Usage

```r
bjobs_barplot(status = c("RUN", "EXIT", "PEND", "DONE"), filter = NULL, df = NULL)
```

### Arguments

- **status**: Status of the jobs. Use "all" for all jobs.
- **filter**: Regular expression to filter on job names.
- **df**: Internally used.

### Details

It draws barplots of number of jobs per day.

### Value

A `ggplot2` object.

### Examples

```r
# There is no example
NULL
```
**bjobs_done**

---

**bjobs_done**  
*Finished jobs*

---

**Description**

Finished jobs

**Usage**

`bjobs_done(max = Inf, filter = NULL)`

**Arguments**

- `max`  
  Maximal number of jobs.
- `filter`  
  Regular expression to filter on job names.

**Details**

You can directly type `bjobs_done` without parentheses which runs `bjobs_done` with defaults.

**Value**

The same output format as `bjobs`.

**Examples**

```r
## Not run:
bjobs_done  # this is the same as `bjobs_done()`
bjobs_done()  # all done jobs
bjobs_done(max = 50)  # last 50 done jobs
bjobs_done(filter = "example")  # done jobs with name ".*example.*"
```

## End(Not run)

---

**bjobs_exit**  
*Failed jobs*

---

**Description**

Failed jobs

**Usage**

`bjobs_exit(max = Inf, filter = NULL)"
### bjobs_pending

**Description**

Pending jobs

**Usage**

```r
bjobs_pending(max = Inf, filter = NULL)
```

**Arguments**

- `max` : Maximal number of jobs.
- `filter` : Regular expression to filter on job names.

**Details**

You can directly type `bjobs_pending` without parentheses which runs `bjobs_exit` with defaults.

**Value**

The same output format as `bjobs`.

---

### bjobs_exit

**Arguments**

- `max` : Maximal number of jobs.
- `filter` : Regular expression to filter on job names.

**Details**

You can directly type `bjobs_exit` without parentheses which runs `bjobs_exit` with defaults.

**Value**

The same output format as `bjobs`.

---

```r
## Not run:
bjobs_exit # this is the same as `bjobs_exit()`
bjobs_exit() # all exit jobs
bjobs_exit(max = 50) # last 50 exit jobs
bjobs_exit(filter = "example") # exit jobs with name ".*example.*"

## End(Not run)
```
bjobs_running

Examples

```r
## Not run:
bjobs_running    # this is the same as `bjobs_pending`
bjobs_pending()  # all pending jobs
bjobs_pending(max = 50) # last 50 pending jobs
bjobs_pending(filter = "example") # pending jobs with name ".*example.*"

## End(Not run)
```

bjobs_running    Running jobs

Description

Running jobs

Usage

```r
bjobs_running(max = Inf, filter = NULL)
```

Arguments

- `max`: Maximal number of jobs.
- `filter`: Regular expression to filter on job names.

Details

You can directly type `bjobs_running` without parentheses which runs `bjobs_running` with defaults.

Value

The same output format as `bjobs`.

Examples

```r
## Not run:
bjobs_running    # this is the same as `bjobs_running`
bjobs_running()  # all running jobs
bjobs_running(max = 50) # last 50 running jobs
bjobs_running(filter = "example") # running jobs with name ".*example.*"

## End(Not run)
```
bjobs_timeline

Timeline of jobs

Description
Timeline of jobs

Usage

bjobs_timeline(status = c("RUN", "EXIT", "PEND", "DONE"), filter = NULL, df = NULL)

Arguments

status Status of the jobs. Use "all" for all jobs.
filter Regular expression to filter on job names.
df Internally used.

Details
It draws segments of duration of jobs. In the plot, each segment represents a job and the width of the segment correspond to its duration.

Value
No value is returned.

Examples

# There is no example
NULL

bkill

Kill jobs

Description
Kill jobs

Usage

bkill(job_id, filter = NULL)

Arguments

job_id A vector of job ids.
filter Regular expression to filter on job names (only the running and pending jobs).
## Value

No value is returned.

## Examples

```r
## Not run:
job_id = c(10000000, 10000001, 10000002)  # job ids can be get from `bjobs`
bkill(job_id)
# kill all jobs (running and pending) of which the names contain "example"
bkill(filter = "example")

## End(Not run)
```

---

### Description

Recent jobs from all status

### Usage

```r
brecent(max = 20, filter = NULL)
```

### Arguments

- `max`: Maximal number of recent jobs.
- `filter`: Regular expression to filter on job names.

### Details

You can directly type `brecent` without parentheses which runs `brecent` with defaults.

### Value

The same output format as `bjobs`.

### Examples

```r
## Not run:
brecht # this is the same as `brecent`
brecent() # last 20 jobs (from all status)
brecht(max = 50) # last 50 jobs
brecent(filter = "example") # last 20 jobs with name ".*example.*"

## End(Not run)
```
bsub_chunk  

Submit R code

Description
Submit R code

Usage
bsub_chunk(code,
    name = NULL,
    packages = bsub_opt$packages,
    image = bsub_opt$image,
    variables = character(),
    share = character(),
    working_dir = bsub_opt$working_dir,
    hours = 1,
    memory = 1,
    cores = 1,
    R_version = bsub_opt$R_version,
    temp_dir = bsub_opt$temp_dir,
    output_dir = bsub_opt$output_dir,
    dependency = NULL,
    enforce = bsub_opt$enforce,
    local = bsub_opt$local,
    script = NULL,
    start = NULL,
    end = NULL,
    save_var = FALSE,
    sh_head = bsub_opt$sh_head)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>The code chunk, it should be embraced by <code>{ }</code>.</td>
</tr>
<tr>
<td>name</td>
<td>If name is not specified, an internal name calculated by digest on the chunk is automatically assigned.</td>
</tr>
<tr>
<td>packages</td>
<td>A character vector with package names that will be loaded before running the script. There is a special name <em>in_session</em> that loads all the packages loaded in current R session.</td>
</tr>
<tr>
<td>image</td>
<td>A character vector of RData/rda files that will be loaded before running the script. When image is set to TRUE, all variables in .GlobalEnv will be saved into a temporary file and all attached packages will be recorded. The temporary files will be removed after the job is finished.</td>
</tr>
<tr>
<td>variables</td>
<td>A character vector of variable names that will be loaded before running the script. There is a special name <em>all_functions</em> that saves all functions defined in the global environment.</td>
</tr>
</tbody>
</table>
bsub_chunk

<table>
<thead>
<tr>
<th>Share</th>
<th>A character vector of variable names for which the variables are shared between jobs. Note the temporary .RData files are not deleted automatically.</th>
</tr>
</thead>
<tbody>
<tr>
<td>working_dir</td>
<td>The working directory.</td>
</tr>
<tr>
<td>hours</td>
<td>Running time of the job.</td>
</tr>
<tr>
<td>memory</td>
<td>Memory usage of the job. It is measured in GB.</td>
</tr>
<tr>
<td>cores</td>
<td>Number of cores.</td>
</tr>
<tr>
<td>R_version</td>
<td>R version.</td>
</tr>
<tr>
<td>temp_dir</td>
<td>Path of temporary folder where the temporary R/bash scripts will be put.</td>
</tr>
<tr>
<td>output_dir</td>
<td>Path of output folder where the output/flag files will be put.</td>
</tr>
<tr>
<td>dependency</td>
<td>A vector of job IDs that current job depends on.</td>
</tr>
<tr>
<td>enforce</td>
<td>If a flag file for the job is found, whether to enforce to rerun the job.</td>
</tr>
<tr>
<td>local</td>
<td>Run job locally (not submitting to the LSF cluster)?</td>
</tr>
<tr>
<td>script</td>
<td>Path of a script where code chunks will be extracted and sent to the cluster. It is always used with start and end arguments.</td>
</tr>
<tr>
<td>start</td>
<td>A numeric vector that contains line indices of the starting code chunk or a character vector that contains regular expression to match the start of code chunks.</td>
</tr>
<tr>
<td>end</td>
<td>Same setting as start.</td>
</tr>
<tr>
<td>save_var</td>
<td>Whether save the last variable in the code chunk? Later the variable can be retrieved by retrieve_var.</td>
</tr>
<tr>
<td>sh_head</td>
<td>Commands that are written as head of the sh script.</td>
</tr>
</tbody>
</table>

Value

Job ID.

See Also

- `bsub_script` submits R scripts.
- `bsub_cmds` submits shell commands.

Examples

```r
## Not run:
bsub_chunk(name = "example", memory = 10, hours = 10, cores = 4,
           {
             Sys.sleep(5)
           })

## End(Not run)
```
**bsub_cmd**

**Submit shell commands**

**Description**

Submit shell commands

**Usage**

```r
bsub_cmd(cmd,
  name = NULL,
  hours = 1,
  memory = 1,
  cores = 1,
  temp_dir = bsub_opt$temp_dir,
  output_dir = bsub_opt$output_dir,
  dependency = NULL,
  enforce = bsub_opt$enforce,
  local = bsub_opt$local,
  sh_head = bsub_opt$sh_head,
  ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>A list of commands.</td>
</tr>
<tr>
<td>name</td>
<td>If name is not specified, an internal name calculated by <code>digest</code> is automatically assigned.</td>
</tr>
<tr>
<td>hours</td>
<td>Running time of the job.</td>
</tr>
<tr>
<td>memory</td>
<td>Memory usage of the job. It is measured in GB.</td>
</tr>
<tr>
<td>cores</td>
<td>Number of cores.</td>
</tr>
<tr>
<td>temp_dir</td>
<td>Path of temporary folder where the temporary R/bash scripts will be put.</td>
</tr>
<tr>
<td>output_dir</td>
<td>Path of output folder where the output/flag files will be put.</td>
</tr>
<tr>
<td>dependency</td>
<td>A vector of job IDs that current job depends on.</td>
</tr>
<tr>
<td>enforce</td>
<td>If a flag file for the job is found, whether to enforce to rerun the job.</td>
</tr>
<tr>
<td>local</td>
<td>Run job locally (not submitting to the LSF cluster)?</td>
</tr>
<tr>
<td>sh_head</td>
<td>Commands that are written as head of the sh script.</td>
</tr>
<tr>
<td>...</td>
<td>Command-line arguments can also be specified as name-value pairs.</td>
</tr>
</tbody>
</table>

**Value**

Job ID.
See Also

- `bsub_chunk` submits R code.
- `bsub_script` submits R scripts.

Examples

```r
## Not run:
bsub_cmd("samtools sort ...", name = ..., memory = ..., cores = ..., ...)
## End(Not run)
```

---

### `bsub_opt`

**Parameters for `bsub`**

**Description**
Parameters for `bsub`

**Usage**

```r
bsub_opt(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)
```

**Arguments**

- `...`: Arguments for the parameters, see "details" section
- `RESET`: reset to default values
- `READ.ONLY`: please ignore
- `LOCAL`: please ignore
- `ADD`: please ignore

**Details**

There are following parameters:

- `packages`: A character vector with package names that will be loaded before running the script.
- `image`: A character vector of RData/rda files that will be loaded before running the script.
- `temp_dir`: Path of temporary folder where the temporary R/bash scripts will be put.
- `output_dir`: Path of output folder where the output/flag files will be put.
- `enforce`: If a flag file for the job is found, whether to enforce to rerun the job.
- `R_version`: The version of R.
- `working_dir`: The working directory.
- `ignore`: Whether ignore `bsub_chunk`, `bsub_script` and `bsub_cmd`.
- `local`: Run job locally (not submitting to the LSF cluster)?
call_Rscript How to call Rscript by specifying an R version number.
submission_node A list of node names for submitting jobs.
login_node This value basically is the same as submission_node unless the login nodes are different from submission nodes.
sh_head Commands that are written as head of the sh script.
user Username on the submission node.
group The user group
ssh_envir The commands for setting bash environment for successfully running bjobs, bsub, ...
bsub_template Template for constructing bsub command.
pars_time A function that parses time string from the LSF bjobs command to a POSIXct object.
verbose Whether to print more messages.

ssh_envir should be properly set so that LSF binaries such as bsub or bjobs can be properly found. There are some environment variables initialized when logging in the bash terminal while they are not initialized with the ssh connection. Thus, some environment variables should be manually set. An example for ssh_envir is as follows. The LSF_ENVDIR and LSF_SERVERDIR should be defined and exported.

```r
c("source /etc/profile",
"export LSF_ENVDIR=/opt/lsf/conf",
"export LSF_SERVERDIR=/opt/lsf/10.1/linux3.10-glibc2.17-x86_64/etc")
```

The values of these two variables can be obtained by entering following commands in your bash terminal (on the submission node):

```
  echo $LSF_ENVDIR
  echo $LSF_SERVERDIR
```

The time strings by LSF bjobs command might be different for different configurations. The R "bsub" package needs to convert the time strings to POSIXlt objects for calculating the time difference. Thus, if the default time string parsing fails, users need to provide a user-defined function and set with parse_time option in bsub_opt. The function accepts a vector of time strings and returns a POSIXlt object. For example, if the time string returned from bjobs command is in a form of Dec 1 18:00:00 2019, the parsing function can be defined as:

```r
  bsub_opt$parse_time = function(x) {
    as.POSIXlt(x, format = "\n  
```

Value

The corresponding option values.

Examples

```
# The default bsub_opt
bsub_opt
```
bsub_script  Submit R script

Description

Submit R script

Usage

bsub_script(script,
            argv = "",
            name = NULL,
            hours = 1,
            memory = 1,
            cores = 1,
            R_version = bsub_opt$R_version,
            temp_dir = bsub_opt$temp_dir,
            output_dir = bsub_opt$output_dir,
            dependency = NULL,
            enforce = bsub_opt$enforce,
            local = bsub_opt$local,
            sh_head = bsub_opt$sh_head,
            ...
)

Arguments

script  The R script.
argv    A string of command-line arguments.
name    If name is not specified, an internal name calculated by digest is automatically assigned.
hours   Running time of the job.
memory  Memory usage of the job. It is measured in GB.
cores   Number of cores.
R_version  R version.
temp_dir Path of temporary folder where the temporary R/bash scripts will be put.
output_dir Path of output folder where the output/flag files will be put.
dependency A vector of job IDs that current job depends on.
enforce   If a flag file for the job is found, whether to enforce to rerun the job.
local    Run job locally (not submitting to the LSF cluster)?
sh_head  Commands that are written as head of the sh script.
...  Command-line arguments can also be specified as name-value pairs.
check_dump_files

Value

Job ID.

See Also

- `bsub_chunk` submits R code.
- `bsub_cmds` submits shell commands.

Examples

```r
## Not run:
bsub_script("/path/of/foo.R", name = ..., memory = ..., cores = ..., ...)
# with command-line arguments
bsub_script("/path/of/foo.R", argv = "--a 1 --b 3", ...)

## End(Not run)
```

check_dump_files Check whether there are dump files

Description

Check whether there are dump files

Usage

```r
check_dump_files(print = TRUE)
```

Arguments

- `print` Whether to print messages.

Details

For the failed jobs, LSF cluster might generate a core dump file and R might generate a .RDataTmp file.
Note if you manually set working directory in your R code/script, the R dump file can be not caught.

Value

A vector of file names.

Examples

```r
## Not run:
check_dump_files()

## End(Not run)
```
clear_temp_dir

Description
Clear temporary dir

Usage
clear_temp_dir(ask = TRUE)

Arguments
ask Whether promote.

Details
The temporary files might be used by the running/pending jobs. Deleting them might affect some of the jobs. You better delete them after all jobs are done.

Value
No value is returned.

Examples
## Not run:
clear_temp_dir()
## End(Not run)

get_dependency

Description
Get the dependency of current jobs

Usage
get_dependency(job_tb = NULL)

Arguments
job_tb A table from bjobs. Optional.
is_job_finished

Value
If there is no dependency of all jobs, it returns NULL. If there are dependencies, it returns a list of three elements:

- **dep_mat**: a two column matrix containing dependencies from parents to children.
- **id2name**: a named vector containing mapping from job IDs to job names.
- **id2stat**: a named vector containing mapping from job IDs to job status.

Examples

```r
## Not run:
get_dependency()

## End(Not run)
```

---

**is_job_finished**  
*Test whether the jobs are finished*

**Description**

Test whether the jobs are finished

**Usage**

```r
is_job_finished(job_name, output_dir = bsub_opt$output_dir)
```

**Arguments**

- **job_name**: A vector of job names.
- **output_dir**: Output dir.

**Details**

It tests whether the "done" flag files exist

**Value**

A logical scalar.

**Examples**

```r
# There is no example
NULL
```
job_log

Log for the running/finished/failed job

Description

Log for the running/finished/failed job

Usage

job_log(job_id, print = TRUE, n_line = 10)

Arguments

job_id The job id. It can be a single job or a vector of job ids.
print Whether print the log message.
n_line Number of last lines for each job to show when multiple jobs are queried.

Value

The log message as a vector.

Examples

## Not run:
# a single job
job_id = 1234567  # job ids can be get from 'bjobs'
job_log(job_id)
# multiple jobs
job_id = c(10000000, 10000001, 10000002)
job_log(job_id)  # by default last 10 lines for each job are printed
job_log(job_id, n_line = 20)  # print last 20 lines for each job
# logs for all running jobs
job_log()

## End(Not run)

job_status_by_id

Job status by id

Description

Job status by id

Usage

job_status_by_id(job_id)
job_status_by_name

Arguments

job_id The job id.

Value

If the job has been deleted from the database, it returns MISSING.

Examples

```r
## Not run:
job_id = 1234567  # job ids can be get from `bjobs`
job_status_by_id(job_id)
## End(Not run)
```

job_status_by_name

Job status by name

Description

Job status by name

Usage

```r
job_status_by_name(job_name, output_dir = bsub_opt$output_dir)
```

Arguments

job_name Job name.

output_dir The output dir.

Value

If the job is finished, it returns DONE/EXIT/MISSING. If the job is running or pending, it returns the corresponding status. If there are multiple jobs with the same name running or pending, it returns a vector.

Examples

```r
## Not run:
job_status_by_name("example")
## End(Not run)
```
monitor

A browser-based interactive job monitor

Description

A browser-based interactive job monitor

Usage

monitor()

Details

The monitor is implemented as a shiny app.

Value

No value is returned.

Examples

## Not run:
# simply run:
monitor
# or
monitor()
## End(Not run)

plot_dependency

Plot the job dependency tree

Description

Plot the job dependency tree

Usage

plot_dependency(job_id, job_tb = NULL)

Arguments

job_id A job ID.
job_tb A table from bjobs. Optional.
Value

No value is returned.

Examples

```r
## Not run:
job1 = random_job()
job2 = random_job()
job3 = random_job(dependency = c(job1, job2))
plot_dependency(job3)

## End(Not run)
```
### Description

Summary of jobs

### Usage

```r
# S3 method for class 'bjobs'
print(x, ...)
```

### Arguments

- **x**: a `bjobs` class object.
- **...**: other arguments.

### Value

No value is returned.

### Examples

```r
# There is no example
NULL
```

---

### random_job

Submit a random job

### Description

Submit a random job

### Usage

```r
random_job(name = paste0("R_random_job_", digest::digest(runif(1), "crc32")), ...)
```

### Arguments

- **name**: Job name.
- **...**: Pass to `bsub_chunk`.

### Details

It only submits `Sys.sleep(30)`.
Value

The job id.

Examples

```r
## Not run:
random_job()
random_job(name = "test")
## End(Not run)
```

---

**retrieve_var**  
Retrieve saved variable

Description

Retrieve saved variable

Usage

```r
retrieve_var(name, output_dir = bsub_opt$output_dir, wait = 30)
```

Arguments

- `name`: Job name.
- `output_dir`: The output dir set in `bsub_chunk`.
- `wait`: Seconds to wait.

Details

It retrieve the saved variable in `bsub_chunk` when `save_rds = TRUE` is set.

Value

The retrieved object.

Examples

```r
## Not run:
bsub_chunk(name = "example", save_var = TRUE,
{
  Sys.sleep(10)
  1+1
})
retrieve_var("example")
## End(Not run)
```
run_cmd

Description
Run command on submission node

Usage
run_cmd(cmd, print = FALSE)

Arguments
- cmd: A single-line command.
- print: Whether to print output from the command.

Details
If current node is not the submission node, the command is executed via ssh.

Value
The output of the command.

Examples
## Not run:
# run pwd on remote node
run_cmd("pwd")
## End(Not run)

ssh_connect

Description
Connect to submission via ssh

Usage
ssh_connect()

Details
If ssh connection is lost, run this function to reconnect.
Value

No value is returned.

Examples

# ssh is automatically connected. To manually connect ssh, run:
## Not run:
ssh_connect()

## End(Not run)
# where the user name is the one you set in `bsub_opt$user` and
# the node is the one you set in `bsub_opt$login_node`.

### ssh_disconnect

**Disconnect ssh connection**

Description

Disconnect ssh connection

Usage

ssh_disconnect()

Value

No value is returned.

Examples

# Normally you don't need to manually run this function. The ssh is automatically
# disconnected when the package is detached.
# To manually disconnect ssh, run:
## Not run:
ssh_disconnect()

## End(Not run)
wait_jobs

Wait until all jobs are finished

Usage

wait_jobs(job_name, output_dir = bsub_opt$output_dir, wait = 30)

Arguments

  job_name     A vector of job names.
  output_dir   Output dir.
  wait         Seconds to wait.

Value

  No value is returned.

Examples

  # There is no example
  NULL
Index

bconf, 2
bjobs, 3, 3, 5–7, 9, 17, 21
bjobs_barplot, 4
bjobs_done, 3, 5, 5
bjobs_exit, 3, 5, 6
bjobs_pending, 3, 6, 6
bjobs_running, 3, 7, 7
bjobs_timeline, 8
bkill, 8
brecent, 3, 9, 9
bsub_chunk, 10, 13, 16, 23, 24
bsub_cmd, 11, 12, 13, 16
bsub_opt, 2, 13, 14
bsub_script, 11, 13, 15
check_dump_files, 16
clear_temp_dir, 17
digest, 10, 12, 15
get_dependency, 17
is_job_finished, 18
job_log, 19
job_status_by_id, 19
job_status_by_name, 20
monitor, 21
plot_dependency, 21
POSIXct, 14
POSIXlt, 14
print.bconf, 22
print.bjobs, 23
random_job, 23
retrieve_var, 11, 24
run_cmd, 25
ssh_connect, 25
ssh_disconnect, 26
wait_jobs, 27