

# Package ‘bsub’

July 22, 2025

**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](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

## Contents

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

bjobs_pending . . . . .	6
bjobs_running . . . . .	7
bjobs_timeline . . . . .	8
bkill . . . . .	8
brecent . . . . .	9
bsub_chunk . . . . .	10
bsub_cmd . . . . .	12
bsub_opt . . . . .	13
bsub_script . . . . .	15
check_dump_files . . . . .	16
clear_temp_dir . . . . .	17
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
print.bconf . . . . .	22
print.bjobs . . . . .	23
random_job . . . . .	23
retrieve_var . . . . .	24
run_cmd . . . . .	25
ssh_connect . . . . .	25
ssh_disconnect . . . . .	26
wait_jobs . . . . .	27

<b>Index</b>	<b>28</b>
--------------	-----------

---

bconf	<i>Print current configuration</i>
-------	------------------------------------

---

## Description

Print current configuration

## Usage

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	<i>Summary of jobs</i>
-------	------------------------

---

**Description**

Summary of jobs

**Usage**

```
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**

```
## 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**

```
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**

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

---

bjobs_done	<i>Finished jobs</i>
------------	----------------------

---

**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**

```
## 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	<i>Failed jobs</i>
------------	--------------------

---

**Description**

Failed jobs

**Usage**

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

**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`.

**Examples**

```
## 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)
```

---

<code>bjobs_pending</code>	<i>Pending jobs</i>
----------------------------	---------------------

---

**Description**

Pending jobs

**Usage**

```
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_pending` with defaults.

**Value**

The same output format as `bjobs`.

## Examples

```
## Not run:
bjobs_pending # 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	<i>Running jobs</i>
---------------	---------------------

---

## Description

Running jobs

## Usage

```
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

```
## 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)
```

---

<code>bjobs_timeline</code>	<i>Timeline of jobs</i>
-----------------------------	-------------------------

---

**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
```

---

<code>bkill</code>	<i>Kill jobs</i>
--------------------	------------------

---

**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**

```
## 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)
```

---

brecent	<i>Recent jobs from all status</i>
---------	------------------------------------

---

**Description**

Recent jobs from all status

**Usage**

```
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**

```
## Not run:
brecent # this is the same as `brecent()`
brecent() # last 20 jobs (from all status)
brecent(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**

code	The code chunk, it should be embraced by { }.
name	If name is not specified, an internal name calculated by <a href="#">digest</a> on the chunk is automatically assigned.
packages	A character vector with package names that will be loaded before running the script. There is a special name <code>_in_session_</code> that loads all the packages loaded in current R session.
image	A character vector of RData/rda files that will be loaded before running the script. When image is set to TRUE, all variables in <code>.GlobalEnv</code> 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.
variables	A character vector of variable names that will be loaded before running the script. There is a special name <code>_all_functions_</code> that saves all functions defined in the global environment.

share	A character vector of variables names for which the variables are shared between jobs. Note the temporary .RData files are not deleted automatically.
working_dir	The working directory.
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)?
script	Path of a script where code chunks will be extracted and sent to the cluster. It is always used with start and end arguments.
start	A numeric vector that contains line indices of the starting code chunk or a character vector that contain regular expression to match the start of code chunks.
end	Same setting as start.
save_var	Whether save the last variable in the code chunk? Later the variable can be retrieved by <a href="#">retrieve_var</a> .
sh_head	Commands that are written as head of the sh script.

### Value

Job ID.

### See Also

- [bsub\\_script](#) submits R scripts.
- [bsub\\_cmd](#) submits shell commands.

### Examples

```
## 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**

```
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**

cmd	A list of commands.
name	If name is not specified, an internal name calculated by <a href="#">digest</a> is automatically assigned.
hours	Running time of the job.
memory	Memory usage of the job. It is measured in GB.
cores	Number of cores.
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.

**Value**

Job ID.

**See Also**

- [bsub\\_chunk](#) submits R code.
- [bsub\\_script](#) submits R scripts.

**Examples**

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

## End(Not run)
```

---

<i>bsub_opt</i>	<i>Parameters for bsub</i>
-----------------	----------------------------

---

**Description**

Parameters for bsub

**Usage**

```
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.  
**parse\_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.

```

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 **\*\*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:

```

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

```

## 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 <a href="#">digest</a> 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.

**Value**

Job ID.

**See Also**

- [bsub\\_chunk](#) submits R code.
- [bsub\\_cmd](#) submits shell commands.

**Examples**

```
## 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**

```
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**

```
## Not run:
check_dump_files()

## End(Not run)
```

---

clear_temp_dir	<i>Clear temporary dir</i>
----------------	----------------------------

---

**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	<i>Get the dependency of current jobs</i>
----------------	-------------------------------------------

---

**Description**

Get the dependency of current jobs

**Usage**

```
get_dependency(job_tb = NULL)
```

**Arguments**

job_tb	A table from <a href="#">bjobs</a> . Optional.
--------	------------------------------------------------

**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**

```
## Not run:
get_dependency()

## End(Not run)
```

---

<code>is_job_finished</code>	<i>Test whether the jobs are finished</i>
------------------------------	-------------------------------------------

---

**Description**

Test whether the jobs are finished

**Usage**

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

**Arguments**

<code>job_name</code>	A vector of job names.
<code>output_dir</code>	Output dir.

**Details**

It tests whether the ".done" flag files exist

**Value**

A logical scalar.

**Examples**

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

---

job_log	<i>Log for the running/finished/failed job</i>
---------	------------------------------------------------

---

**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	<i>Job status by id</i>
------------------	-------------------------

---

**Description**

Job status by id

**Usage**

```
job_status_by_id(job_id)
```

**Arguments**

job\_id            The job id.

**Value**

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

**Examples**

```
## 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	<i>Job status by name</i>
--------------------	---------------------------

---

**Description**

Job status by name

**Usage**

```
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**

```
## Not run:
job_status_by_name("example")

## End(Not run)
```

---

monitor	<i>A browser-based interactive job monitor</i>
---------	------------------------------------------------

---

**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	<i>Plot the job dependency tree</i>
-----------------	-------------------------------------

---

**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 <a href="#">bjobs</a> . Optional.

**Value**

No value is returned.

**Examples**

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

## End(Not run)
```

---

print.bconf	<i>Print the configurations</i>
-------------	---------------------------------

---

**Description**

Print the configurations

**Usage**

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

**Arguments**

x	A bconf object
...	Other parameters

**Value**

No value is returned.

**Examples**

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

---

print.bjobs	<i>Summary of jobs</i>
-------------	------------------------

---

**Description**

Summary of jobs

**Usage**

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

**Arguments**

x	a bjobs class object.
...	other arguments.

**Value**

No value is returned.

**Examples**

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

---

random_job	<i>Submit a random job</i>
------------	----------------------------

---

**Description**

Submit a random job

**Usage**

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

**Arguments**

name	Job name.
...	Pass to <a href="#">bsub_chunk</a> .

**Details**

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

**Value**

The job id.

**Examples**

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

## End(Not run)
```

---

retrieve_var	<i>Retrieve saved variable</i>
--------------	--------------------------------

---

**Description**

Retrieve saved variable

**Usage**

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

**Arguments**

name	Job name.
output_dir	The output dir set in <a href="#">bsub_chunk</a> .
wait	Seconds to wait.

**Details**

It retrieve the saved variable in [bsub\\_chunk](#) when save\_rds = TRUE is set.

**Value**

The retrieved object.

**Examples**

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

## End(Not run)
```

---

`run_cmd`*Run command on submission node*

---

**Description**

Run command on submission node

**Usage**

```
run_cmd(cmd, print = FALSE)
```

**Arguments**

<code>cmd</code>	A single-line command.
<code>print</code>	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`*Connect to submission via ssh*

---

**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	<i>Disconnect ssh connection</i>
----------------	----------------------------------

---

**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	<i>Wait until all jobs are finished</i>
-----------	-----------------------------------------

---

**Description**

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](#)