

# Package ‘job’

June 4, 2021

**Title** Run Code as an RStudio Job - Free Your Console

**Version** 0.3.0

**Description** Call `job::job({<code here>})` to run R code as an RStudio job and keep your console free in the meantime. This allows for a productive workflow while testing (multiple) long-running chunks of code. It can also be used to organize results using the RStudio Jobs GUI or to test code in a clean environment. Two RStudio Addins can be used to run selected code as a job.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Language** en-US

**Depends** R (>= 3.5.0)

**Imports** rstudioapi (>= 0.13), digest (>= 0.6.27)

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Jonas Kristoffer Lindeløv [aut, cre]  
(<<https://orcid.org/0000-0003-4565-0595>>)

**Maintainer** Jonas Kristoffer Lindeløv <jonas@lindeloev.dk>

**Repository** CRAN

**Date/Publication** 2021-06-04 06:00:02 UTC

## R topics documented:

export . . . . .	2
job . . . . .	3
print.jobcode . . . . .	5
<b>Index</b>	<b>6</b>

---

 export

*What to return from a job*


---

### Description

Call this function as the last line in `job::job()` to select what is exported back into `globalenv()`. `export()` does nothing if called in any other context.

### Usage

```
export(value = "changed", file = NULL)
```

### Arguments

value	What to return. One of: <ul style="list-style-type: none"> <li>• "all": Return everything, including imports</li> <li>• "changed" (default): Return all variables that are not identical to import.</li> <li>• "new": Return only new variable names.</li> <li>• <code>c(var1, var2, ...)</code>: Return these variable names.</li> <li>• NULL or "none": Return nothing. This is particularly useful for unnamed code chunks.</li> </ul>
file	Name of .RData file to export to. If not NULL, nothing will be returned to the main session (corresponding to <code>export("none")</code> ).

### Details

Under the hood, this function merely `rm()` variables that does not match `value`. Because `job::job()` returns everything at the end of the script, this defines what is returned.

### Value

NULL invisibly.

### Author(s)

Jonas Kristoffer Lindeløv, <jonas@lindeloev.dk>

### Examples

```
if (rstudioapi::isAvailable()) {
  a = 55
  b = 77
  d = 88
  job::job({n = 11; a = 55; job::export("all")}) # export a, b, d, n
  job::job({n = 11; a = 11; job::export("changed")}) # export a, n
  job::job({n = 11; a = 11; job::export("new")}) # export n
  job::job({n = 11; a = 55; job::export(c(a, d, b))}) # export a, d, b
}
```

```

job::job({n = 11; a = 55; job::export("none")}) # export nothing

# To file
job::job({n = 11; a = 11; job::export("changed", file = "jobresult.RData")}) # save a, n
jobresult = new.env() # import to this env instead of global
load("jobresult.RData", envir = jobresult)
print(jobresult$n)
}

```

---

 job

*Run Code as an RStudio Job*


---

## Description

See examples for an introduction. See [the job website](#) for more examples. See details for some warnings. Note that `job::empty()` is identical to `job::job()` but all arguments default to `NULL`.

## Usage

```

job(
  ...,
  import = "all",
  packages = .packages(),
  opts = options(),
  title = NULL
)

empty(..., import = NULL, packages = NULL, opts = NULL, title = NULL)

```

## Arguments

...	A named or unnamed code block enclosed in curly brackets, <code>{}</code> . Named code blocks will assign the that name in <code>globalenv()</code> . Unnamed code blocks will assign job variables directly to <code>globalenv()</code> upon completion. Control what gets returned using <code>export</code> within the code block.
import	Which objects to import into the job. <ul style="list-style-type: none"> <li>• "all": Import all objects.</li> <li>• "auto" (default): Detect which objects are used in the code and import those.</li> <li>• <code>c(foo, bar, ...)</code>: A vector of unquoted variables to import into the job.</li> <li>• <code>NULL</code>: import nothing.</li> </ul>
packages	Character vector of packages to load in the job. Defaults to all loaded packages in the calling environment. <code>NULL</code> loads only default packages. You can combine <code>packages = NULL</code> with writing <code>library(my_package)</code> in the code block.
opts	List of options to overwrite in the job. Defaults to <code>options()</code> , i.e., copy all options to the job. <code>NULL</code> uses defaults.

`title` The job title. You can write e.g., "Cross-Validation: {code}" to include a code snippet in the title. If `title = NULL` (default), the name of the code chunk is used. If `...` is unnamed, the code is shown.

### Details

This is a wrapper around `rstudioapi::jobRunScript`. To control what gets returned, see [export](#). By default, all objects that *changed* during the job are returned, i.e., `job::export("changed")`.

- **Returning large objects:** `jobRunScript` is very slow at importing and exporting large objects. For exporting back into `globalenv()`, it may be faster to `saveRDS()` results within the job and `readRDS()` them in your environment.

### Value

Invisibly returns the job id on which you can call other `rstudioapi::job*` functions, e.g., `rstudioapi::rstudioapi::jobRemove(job)`.

### Functions

- `empty`: `job::job()` but with `NULL` defaults, i.e., an "empty" job.

### Author(s)

Jonas Kristoffer Lindeløv, <jonas@lindeloev.dk>

### See Also

[export](#), [jobRunScript](#)

### Examples

```
if (rstudioapi::isAvailable()) {
  # Unnamed code chunks returns to globalenv()
  global_var = 5
  job::job({
    x = rnorm(global_var)
    print("This text goes to the job console")
    m = mean(x)
  })

  # later:
  print(x)
  print(m)

  # Named code chunks assign job environment to that name
  job::job(my_result = {
    y = rnorm(global_var)
    sigma = sd(y)
  }, title = "Title with code: {code}")

  # later:
```

```

print(my_result$y)
print(my_result$sigma)

# Delete everything in the job environment to return nothing.
# Useful if text output + file output is primary
job::job({
  some_cars = mtcars[mtcars$cyl > 4, ]
  print(mean(some_cars$mpg))
  print(summary(some_cars))
  # saveRDS(some_cars, "job_result.rds")

  job::export("none") # return nothing
})

# Control imports from calling environment (variables, packages, options)
my_df = data.frame(names = c("alice", "bob"))
ignore_var = 15
job::job(result2 = {
  if (exists("ignore_var") == FALSE)
    print("ignore_var is not set here")

  names = rep(my_df$names, global_var)
}, import = c(global_var, my_df), packages = NULL, opts = list(mc.cores = 3))

# later
print(result2$names)
}

```

---

```
print.jobcode      Nice print.jobcode
```

---

## Description

Nice print.jobcode

## Usage

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

## Arguments

x	Text to print
...	Currently unused

## Value

No return value, called for side effects.

# Index

`empty (job)`, 3  
`export`, 2, 3, 4

`job`, 3  
`jobRunScript`, 4

`print.jobcode`, 5