Schedule R scripts with taskscheduleR

Let’s install **taskscheduleR** using the *install.packages* command.

```
install.packages("taskscheduleR")
```

Next, we just need to load the package to get started.

```
library(taskscheduleR)
```

Creating a sample R script to run automatically

Before we do any scheduling, we need to first create a script. We’ll save the code below in a file called “create_file.txt”. This script will randomly generate a collection of integers and write them out to a file.

```
nums <- sample(10000, 100)

write(nums, "sample_nums.txt")
```

Using the taskscheduler_create function

Next, in order to schedule the script to run automatically, we need to use the *taskscheduler_create* function. This function takes several arguments, which can be seen below.

```
taskscheduler_create(taskname = "test_run", rscript = "/path/to/file
/create_file.R",
                     schedule = "ONCE", starttime = format(Sys.time() + 50,
"%H:%M"))
```

Firstly, we need to give a name to the task we want to create. In this case, we’ll just call our task “test_run”. Next, we need to specify the R script we want to automatically run. Third, we add the *schedule* parameter, which denotes how frequently we want to run this script. There are several options here, including WEEKLY, DAILY, MONTHLY, HOURLY, and MINUTE. For example, if we want our script to run every day, we would modify our function call like this:

```
taskscheduler_create(taskname = "test_run", rscript = "/path/to/file
/create_file.R",
                     schedule = "DAILY", starttime = format(Sys.time() + 50,
"%H:%M"))
```

The other parameter we need to select is start time. In our examples, we’re setting the task to start in 50 seconds from the current time.

In addition to these arguments, *taskscheduler_create* also has a parameter called “modifier”. This allows us to modify the schedule frequency. For example, what if we want to run the task every 2 hours? In this case, we would just set modifier = 2.

```
taskscheduler_create(taskname = "test_run", rscript = "/path/to/file
/create_file.R",
                     schedule = "HOURLY", starttime = format(Sys.time() + 50,
"%H:%M"), modifier = 2)
```

.

Similarly, we could run our script every 10 minutes using the code below, with a modifier of 10.

```
taskscheduler_create(taskname = "test_run", rscript = "/path/to/file
/create_file.R",
                     schedule = "MINUTE", starttime = format(Sys.time() + 50,
"%H:%M"), modifier = 10)
```

Passing arguments to the Task Scheduler

We can pass arguments to the scheduled task using the “rscript_args” parameter.

```
taskscheduler_create(taskname = "test_run", rscript = "/path/to/file
/create_file.R",
                     schedule = "MINUTE", starttime = format(Sys.time() + 50,
"%H:%M"), modifier = 10,
                     rscript_args = c("10", "test", "this"))
```

Listing the tasks in the scheduler

What if we want to look at all the tasks currently in the task scheduler? There’s a quick method for that called *taskscheduler_ls*.

```
taskscheduler_ls()
```

Running this function returns a data frame of all the tasks currently in the task scheduler.

Deleting tasks from the scheduler

Tasks can be deleted from the scheduler using the *taskscheduler_delete* function. We just need to input the the name of the task.

```
taskscheduler_delete("test_run")
```

Stopping tasks

Similarly, tasks can also be stopped with a single function call. This time, we’ll use the *taskscheduler_stop* function.

```
taskscheduler_stop("test_run")
```

Scheduling tasks on Linux

If you're running on Linux, there's an alternative package called **cronR** which can be used. To use this package on Linux, we need to install it using **devtools**, like below.

```
devtools::install_github("bnosac/cronR")
```

Additionally, the cron daemon needs to be running for the package to work properly. On Debian, you can ensure this by running the below commands:

```
sudo apt-get update
sudo apt-get install -y cron
sudo cron start
```

Schedule R scripts on Linux

Once installed, we can get started creating tasks. In this example we create a task that runs daily, starting at 9:00 AM.

```
library(cronR)

cmd <- cron_rscript("/path/to/file/create_file.R")

cron_add(command = cmd, frequency = 'daily', at = "9:00" , id =
'test_linux_run', description = "testing linux scheduler")
```

To change the time, we just need to adjust the "at" parameter. The example below runs the same script automatically at 3:00 PM (15:00) each day.

```
cron_add(command = cmd, frequency = 'daily', at = "15:00" , id =
'test_linux_run', description = "testing linux scheduler")
```

The next example runs the same task every 30 minutes.

```
cron_add(command = cmd, frequency = '/30 * * * *', at = "15:00" , id =
'test_linux_run', description = "testing linux scheduler")
```

Passing arguments to cronR

Running an R script with command line arguments is a common need. This can be handled with **cronR** using the "rscript_args" parameter in the *cron_rscript* function.

```
cmd <- cron_rscript("/path/to/file/create_file.R", rscript_args = c("10",
"test", "this"))

cron_add(command = cmd, frequency = '/30 * * * *', at = "15:00" , id =
'test_linux_run', description = "testing linux scheduler")
```

Running cron jobs on specific days of the week

It’s also possible to select what days of the week you want a job to run. For example, if we want a job to run hourly, but only on certain days of the week, we could specify that like below.

```
cron_add(command = cmd, frequency = 'hourly', at = "15:00" , id =
'test_linux_run', description = "testing linux scheduler",
         days_of_week = c(0,3))
```

Running cron jobs on specific days of the month

Similarly, we can also select days of the month in which to run jobs.

```
cron_add(command = cmd, frequency = 'hourly', at = "15:00" , id =
'test_linux_run', description = "testing linux scheduler",
         days_of_month = c(10, 20, 30))
```

Listing all Linux tasks

cronR can list all scheduled Linux jobs using the *cron_ls* function.

```
cron_ls()
…
```