

# How to use R on the Bioinformatics cluster

Gaëlle Lefort & Nathalie Vialaneix

July 21, 2022

*Dernière mise à jour : 21 juillet 2022*

**DO NOT run treatments on frontal servers, always use sbatch or srun.**  
**Before contacting the support, READ THE FAQ:** <http://bioinfo.genotoul.fr/index.php/faq/>.

This tutorial aims at describing how to run R scripts and compile RMarkdown files on the Toulouse Bioinformatics<sup>1</sup> cluster. To do so, you need to have an account (ask for an account on this page <http://bioinfo.genotoul.fr/index.php/ask-for/create-an-account/>). You can then connect to the cluster using the `ssh` command on linux and Mac OS and using Putty<sup>2</sup> on Windows. Similarly, you can copy files between the cluster and your computer using the `scp` command on linux and Mac OS and using WinSCP<sup>3</sup> on Windows. The login address is `genologin.toulouse.inra.fr`. Once you are connected, you have two solutions to run a script: running it in batch mode or starting an interactive session. The script must never be run on the first server you connect to. Also, be careful that the programs that you can use from the cluster are not available until you have loaded the corresponding module. How to manage modules is explained in Section 1.

## Contents

<b>1 Use of modules</b>	<b>1</b>
<b>2 Run an R script in batch mode</b>	<b>2</b>
2.1 <code>sbatch</code> options	2
2.2 Job management	3
<b>3 Use R in an interactive mode</b>	<b>3</b>
3.1 X11 sessions	3
<b>4 R in a parallel environment</b>	<b>4</b>
4.1 Parallel with <code>doParallel</code> package	4
<b>5 Arguments in a script</b>	<b>5</b>
<b>6 Install packages in your own environment</b>	<b>5</b>
<b>7 Create and compile <code>.Rmd</code> (RMarkdown) files on the cluster (batch mode)</b>	<b>6</b>

## 1 Use of modules

All programs are made available by loading the corresponding module. These commands are the main useful commands to work with modules:

---

<sup>1</sup><http://bioinfo.genotoul.fr/>

<sup>2</sup><https://putty.org/>

<sup>3</sup><https://winscp.net/eng/index.php>

- `module avail`: list all available modules
- `search_module [TEXT]`: find a module with keyword
- `module load [MODULE_NAME]`: to load a module (for instance to load R `module load system/R-3.5.1`). This command is either used directly (in interactive mode) or included in the file that is used to run your R script in batch mode (see below)
- `module purge`: purge all previous loaded modules

## 2 Run an R script in batch mode

To launch an R script on the slurm cluster:

1. Write an R script:

Script 1: HelloWorld.R

```
1 print("Hello world!")
```

2. Write a bash script:

Script 2: myscript.sh

```
1  #!/bin/bash
2  #SBATCH -J launchRscript
3  #SBATCH -o output.out
4
5  # Purge all previously loaded modules
6  module purge
7
8  # Load the R module
9  module load system/R-3.5.1
10
11 # The command lines that I want to run on the cluster
12 Rscript HelloWorld.R
```

3. Launch the (bash) script with the `sbatch` command:

```
1 sbatch myscript.sh
```

The scripts `myscript.sh` and `HelloWorld.R` are supposed to be located in the same directory from which the `sbatch` command is launched. For Rmd files (see section 7), be careful that you cannot compile a document if the Rmd file is not in a writable directory.

### 2.1 sbatch options

Jobs can be launched with customized options (more memory, for instance). There are two ways to handle `sbatch` options:

- **[RECOMMENDED]** at the beginning of the bash script with lines of the form: `#SBATCH [OPTION] [VALUE]`
- in the `sbatch` command: `sbatch [OPTION] [VALUE] [OPTION] [VALUE] [...] myscript.sh`

Many options are available. To see all options use `sbatch --help`. Useful options:

- `-J, --job-name=jobname`: name of job
- `-e, --error=err`: file for batch script's standard error

- `-o, --output=out`: file for batch script's standard output
- `--mail-type=BEGIN,END,FAIL`: send an email at beginning, end or fail of the script (default email is your user email and can be changed with `--mail-user=truc@bidule.fr`, to use with care)
- `-t, --time=HH:MM:SS`: time limit (default to 04:00:00)
- `--mem=XG`: to change memory reservation (default to 4G)
- `-c, --cpus-per-task=ncpus`: number of cpus required per task (default to 1)
- `--mem-per-cpu=XG`: maximum amount of real memory per allocated cpu required by the job

## 2.2 Job management

After a job has been launched, you can monitor it with `squeue -u [USERNAME]` or `squeue -j [JOB.ID]` and also cancel it with `scancel [JOB.ID]`.

## 3 Use R in an interactive mode

To use R in a console mode, use `srun --pty bash` to be connected to a node. Then, `module load system/R-3.5.1` (for the last R version) and `R` to launch R.

```
@genologin2 ~/work $ srun --pty bash
@node137 ~/work $ module load system/R-3.5.1
@node137 ~/work $ R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> 
```

`srun` can be run with the same options than `sbatch` (cpu and memory reservations) (see section 2.1).

### 3.1 X11 sessions

X11 sessions are useful to directly display plots in an interactive session. Prior their use, generate a ssh key with `ssh-keygen` and add it in authorized keys file with

```
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

The interactive session is then launched with:

1. Logging on the cluster with `ssh -Y [USERNAME]@genologin.toulouse.inra.fr`
2. Run an interactive session with `srun --x11 --pty bash`

```

@genologin2 ~ $ srun --x11 --pty bash
@node126 ~ $ module load system/R-3.5.1
@node126 ~ $ R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> plot(1:10)
>

```

## 4 R in a parallel environment

To use R with a parallel environment, the `-c`, `--cpus-per-task=ncpus` option for the `sbatch` and `srun` is needed. In the R script, the SAME number of cores needs to be specified.

### 4.1 Parallel with `doParallel` package

Several packages exist to use parallel with R: `doParallel`, `BiocParallel`, and `future` (examples are provided for 2 parallel jobs and the two first packages).

1. Write an R script:

- With `doParallel` package:

Script 3: TestParallel.R

```

1 library(doParallel)
2 # specify the number of cores with makeCluster
3 cl <- makeCluster(2)
4 registerDoParallel(cl)
5
6 foreach(i = 1:3) %dopar% sqrt(i)

```

- With `BiocParallel` package:

Script 4: TestParallel.R

```

1 library(BiocParallel)
2
3 # specify the number of cores with workers = 2
4 bplapply(1:10, print, BPPARAM = MulticoreParam(workers = 2))

```

2. Write a bash script:

Script 5: myscript.sh

```

1 #!/bin/bash
2 #SBATCH -J launchRscript
3 #SBATCH -o output.out
4 #SBATCH -c 2
5

```

```

6      #Purge any previous modules
7      module purge
8
9      #Load the application
10     module load system/R-3.5.1
11
12     # My command lines I want to run on the cluster
13     Rscript TestParallel.R

```

3. Launch the script with the `sbatch` command:

```
1      sbatch myscript.sh
```

## 5 Arguments in a script

External arguments can be passed to an R script. The basic method is described below but the package `optparse` provides ways to handle external arguments *à la* Python.

1. Write an R script:

Script 6: HelloWorld.R

```

1      args <- commandArgs(trailingOnly=TRUE)
2
3      print(args[1])

```

2. Write a bash script:

Script 7: myscript.sh

```

1      #! /bin/bash
2      #SBATCH -J launchRscript
3      #SBATCH -o output.out
4
5      #Purge any previous modules
6      module purge
7
8      #Load the application
9      module load system/R-3.5.1
10
11     # My command lines I want to run on the cluster
12     Rscript --vanilla HelloWorld.R "Hi!"

```

3. Launch the script with the `sbatch` command:

```
1      sbatch myscript.sh
```

## 6 Install packages in your own environment

Once in an interactive R session, R packages are installed (in a personal library) using the standard `install.packages` command line.

```

@genologin2 ~/work $ srund --pty bash
@node126 ~/work $ export R_LIBS=~/.work/Lib"
@node126 ~/work $ module load system/R-3.5.1
@node126 ~/work $ R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> install.packages("ggplot2")

```

Your personal library is usually located at the root of your personal directory whose allocated space is very limited. A simple solution consists in:

1. creating a directory named R elsewhere: `mkdir ~/.work/R`
2. making a symbolic link to this directory: `ln -s ~/.work/R ~/R`

## 7 Create and compile .Rmd (RMarkdown) files on the cluster (batch mode)

To compile an .Rmd file, two packages are needed: `rmarkdown` and `knitr`. You also need to load the module `system/pandoc-2.1.3`.

As for an R script, you can pass external arguments to a .Rmd document.

1. Write an .Rmd script with parameters in the header:

Script 8: MyDocument.Rmd

```

1  ---
2  title: My Document
3  output: html_document
4  params:
5     text: "Hi!"
6  ---
7
8  What is your text ?
9  ““{r}
10 print(params$text)
11 ““

```

2. Write an R script to pass parameters:

Script 9: TestRmd.R

```

1  rmarkdown::render("MyDocument.Rmd", params = list(text = "Hola!"))

```

3. Write a bash script:

Script 10: myscript.sh

```
1  #!/bin/bash
2  #SBATCH -J launchRscript
3  #SBATCH -o output.out
4
5  module purge
6  module load system/R-3.5.1
7  module load system/pandoc-2.1.3
8
9  Rscript --vanilla TestRmd.R
```

4. Launch the script with the `sbatch` command:

```
1  sbatch myscript.sh
```