How to use R on the Bioinformatics cluster

Authors: Gaëlle Lefort, Alyssa Imbert, Julien Henry & Nathalie Vialaneix

Last update: October 2023

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[1] 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[2] 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[3] on Windows. The login address is `genobioinfo.toulouse.inrae.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 been 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

1 Use of modules

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

[1] `http://bioinfo.genotoul.fr/`
[2] `https://putty.org/`
[3] `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 statistics/R/4.3.0`). 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

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

2. Write a `bash` script:

Script 2: myscript.sh

```
#! /bin/bash
#SBATCH -J lauchRscript
#SBATCH -o output.out

# Purge all previously loaded modules
module purge

# Load the R module
module load statistics/R/4.3.0

# The command lines that I want to run on the cluster
Rscript HelloWorld.R
```

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

```
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

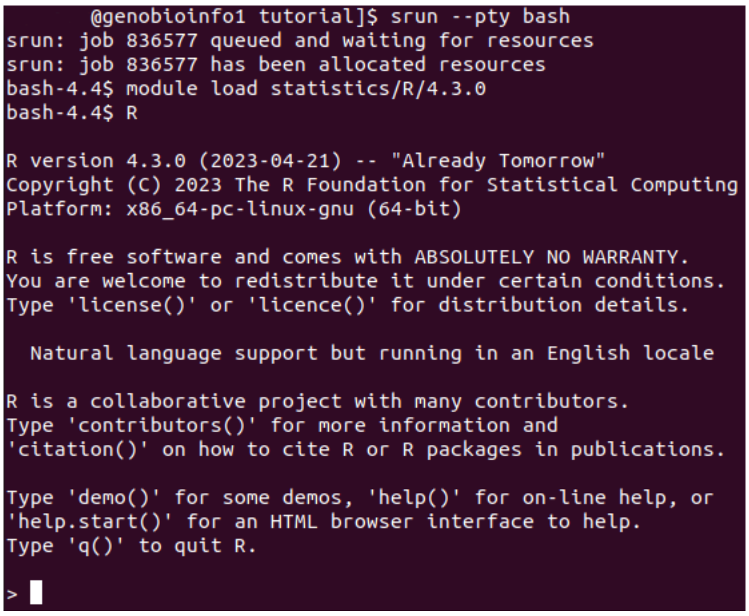
- `-o, --output=out`: file for batch script's standard output
- `--mail-type=BEGIN,END,FAIL`: send an email at the beginning, end or fail of the script (default email is your user email and can be changed with `--mail-user=truc@bidule.fr`, to use with care)
- `-t, --time=HH:MM:SS`: time limit (default 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 interactive mode

To use R in a console mode, use `srun --pty bash` to be connected to a node. Then, `module load statistics/R/4.3.0` (for the latest R version) and `R` to launch R.



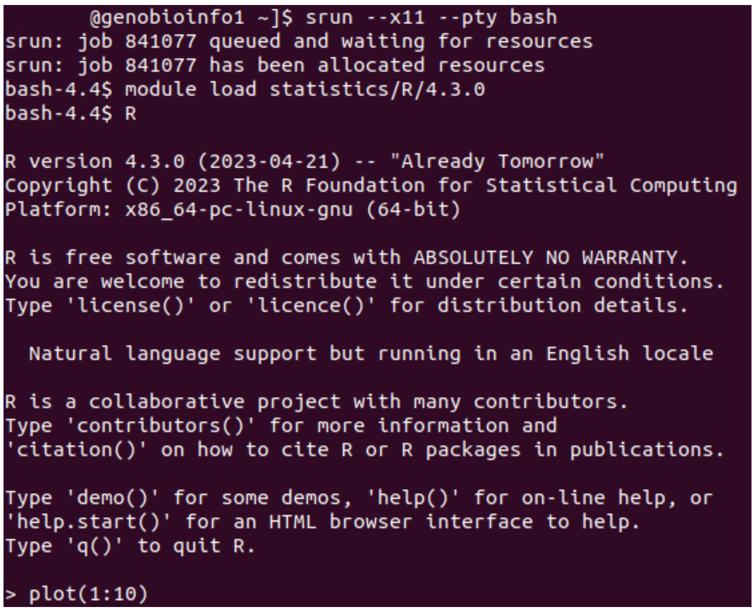
```
       @genobioinfo1 tutorial]$ srun --pty bash
srun: job 836577 queued and waiting for resources
srun: job 836577 has been allocated resources
bash-4.4$ module load statistics/R/4.3.0
bash-4.4$ R

R version 4.3.0 (2023-04-21) -- "Already Tomorrow"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit 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 an 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 by:

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



```
      @genobioinfo1 ~]$ srun --x11 --pty bash
srun: job 841077 queued and waiting for resources
srun: job 841077 has been allocated resources
bash-4.4$ module load statistics/R/4.3.0
bash-4.4$ R

R version 4.3.0 (2023-04-21) -- "Already Tomorrow"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> plot(1:10)
```

4 R in a parallel environment

To use R with a parallel environment, the `-c` (or `--cpus-per-task`) option for the `sbatch` and `srun` is needed. In the R script, the number of cores must be set to the SAME value.

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 first two packages).

1. Write an R script:

   - With **doParallel** package:

Script 3: TestParallel.R

```r
library(doParallel)
# specify the number of cores with makeCluster
cl <- makeCluster(2)
registerDoParallel(cl)

foreach(i=1:3) %dopar% sqrt(i)
```

   - With **BiocParallel** package:

Script 4: TestParallel.R

```r
library(BiocParallel)

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

2. Write a `bash` script:

Script 5: myscript.sh

```bash
#! /bin/bash
#SBATCH -J lauchRscript
```

```
#SBATCH -o output.out
#SBATCH -c 2

#Purge any previous modules
module purge

#Load the application
module load statistics/R/4.3.0

# My command lines I want to run on the cluster
Rscript TestParallel.R
```

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

```
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

```
args <- commandArgs(trailingOnly=TRUE)

print(args[1])
```

2. Write a `bash` script:

Script 7: myscript.sh

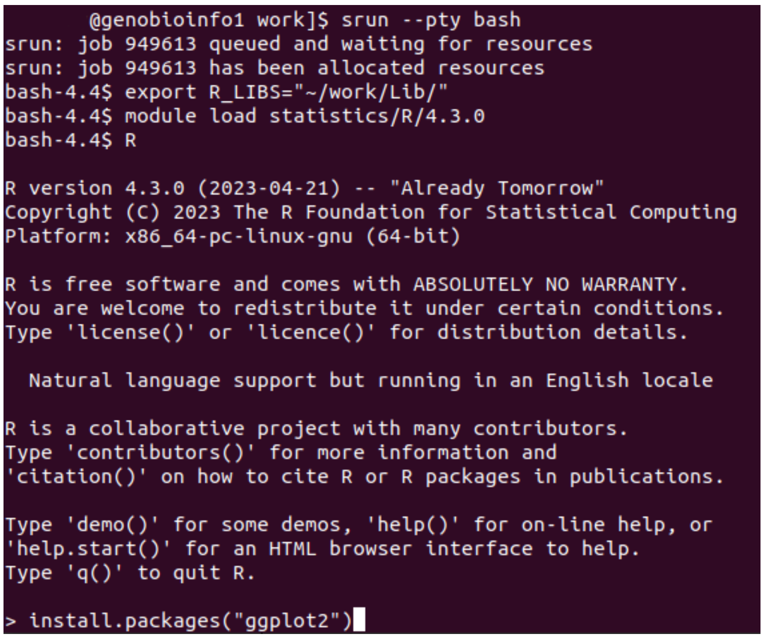
```
#! /bin/bash
#SBATCH -J lauchRscript
#SBATCH -o output.out

#Purge any previous modules
module purge

#Load the application
module load statistics/R/4.3.0

# My command lines I want to run on the cluster
Rscript --vanilla HelloWorld.R "Hi!"
```

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

```
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.



```
       @genobioinfo1 work]$ srun --pty bash
srun: job 949613 queued and waiting for resources
srun: job 949613 has been allocated resources
bash-4.4$ export R_LIBS="~/work/Lib/"
bash-4.4$ module load statistics/R/4.3.0
bash-4.4$ R

R version 4.3.0 (2023-04-21) -- "Already Tomorrow"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("ggplot2")
```

Your personal library is usually located at the root of your personnal directory whose allocated space is very limited. A simple solution consist 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 a `.Rmd` file, two packages are needed: **rmarkdown** and **knitr**. You also need to load the module `tools/Pandoc/3.1.2`.

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

1. Write a `.Rmd` script with parameters in the header:

Script 8: MyDocument.Rmd

~~~
---
title: My Document
output: html_document
params:
    text: "Hi!"
---

What is your text?
```{r}
print(params$text)
```
~~~

2. Write an R script to pass parameters:

Script 9: TestRmd.R

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

3. Write a `bash` script:

Script 10: myscript.sh

```bash
#! /bin/bash
#SBATCH -J lauchRscript
#SBATCH -o output.out

module purge
module load statistics/R/4.3.0
module load tools/Pandoc/3.1.2

Rscript --vanilla TestRmd.R
```

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

```bash
sbatch myscript.sh
```