UNIVERSITY
OF OSLO

R

Table of Contents

Description

R is a software environment for statistical computing and graphics. See R's home page and documentation for details.

Home page

http://www.r-project.org/

Documentation

http://cran.r-project.org/manuals.html

License

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.

Usage

Use

```
module avail R
```

to see which versions of R are available. Use

```
module load R/version
```

to get access to R.

The best way of running R in a script, is to use

```
Rscript yourscript.R
```

or, if you do **not** want R to read and save .RData:

```
Rscript --no-save --no-restore yourscript.R
```

For details see:

```
Rscript --help
```

Installed packages

Several packages have been pre-installed on Colossus. Do

```
library()
```

in R to see what is available. If you start R with any of the switches `--vanilla` or `--no-environ`, you will *not* have access to the pre-installed packages.

Installing packages

We encourage you to install the packages you need yourself. That way you can upgrade packages whenever you need. You can install packages in R with:

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

In TSD, Linux hosts will automatically retrieve packages from our internal CRAN/Bioconductor mirrors as outlined here.

Checkpointing

If you run calculations that take more than a day, it is usually a good idea to use checkpointing, i.e, save the state of the calculations at intervals and use these saved values whenever the script is restarted. For an iteration based simulation, checkpointing can be implemented with something like this:

```
## Load any R packages
## ...

## Load the checkpointing file, if it exists:
checkFile <- "checkpoint.RData"
tempFile <- "tempCheckpoint.RData"
if (file.exists(checkFile)) {
 load(checkFile)
 cat("Resuming after iteration", iter, "\n")
 iter <- iter + 1                      # Important to avoid infinite loop
} else {
 ## Set up the simulation
 ## ...

 Nsims <- 1000
 iter <- 1                             # The first iteration
}

## Do the simulation
if (iter <= Nsims) {                    # In case we were interrupted after
                                       # the last iteration
 for (iter in iter:Nsims) {            # Note _iter_:Nsims
   cat("Iteration", iter, "\n")
   ## Do iteration iter
   ## ...

   ## Save the results of the iteration.  (By first saving to a temporary file
   ## and then renaming it into the checkpointing file, we guard against being
   ## interrupted while saving.)
   save.image(tempFile)
   file.rename(tempFile, checkFile)
 }
}

## Do final stuff and save results
## ...

## Clean up (_after_ saving the results)
if (file.exists(checkFile)) file.remove(checkFile)
```

Parallel jobs

There are many ways of parallelising jobs in R. Since version 2.14.0, R comes with a package called `parallel`, which provides perhaps the simplest interface for setting up parallel jobs. The following is a simple illustration of how to use the package to run a calculation as an MPI job on Colossus:

R script (test_parallel_MPI.R):

```
library(parallel)

## Start the MPI worker processes:
numWorkers <- as.numeric(Sys.getenv("SLURM_NTASKS")) - 1
myCluster <- makeCluster(numWorkers, type = "MPI")

## If needed, load any libraries, etc. on the workers:
# Run library(car) on all workers:
#clusterEvalQ(myCluster, library(car))
# Export variables var1 and var2 to all workers:
#clusterExport(myCluster, c("var1", "var2"))

## Define a worker function:
workerFunc <- function(n) {
    return(n^2)
}
## and a list or vector to use it on:
values <- 1:100

## Apply workerFunc to values in parallel:
results <- parLapply(myCluster, values, workerFunc)
print(unlist(results))

## Exit cleanly:
stopCluster(myCluster)
Rmpi::mpi.finalize()
# or Rmpi::mpi.quit(), which quits R as well
# (Note that "mpi.exit()" and "mpi.quit()" no longer works.)
```

Slurm script:

```
#!/bin/bash

#SBATCH --ntasks=NumberOfProcesses
#SBATCH --account=MyAccount --time=hh:mm:ss --mem-per-cpu=megabytes

module load R/4.0.0-foss-2020a

mpirun -n 1 R --slave < test_parallel_MPI.R
```

In order to learn more about the `parallel` package, or parallel R jobs in general, we suggest reading the vignette that comes with the `parallel` package, or the book *Parallel R* by Q. Ethan McCallum and Stephen Weston.

Published June 21, 2021 10:35 AM - Last modified July 6, 2022 3:41 PM