randomForestSRC: Parallel Processing Vignette

Hemant Ishwaran, Min Lu and Udaya B. Kogalur

OpenMP Parallel Processing – Setting the Number of CPUs

There are several ways to control the number of CPU cores that the package accesses during OpenMP parallel execution. First, you will need to determine the number of cores on your local machine. Do this by starting an R session and issuing the command `detectCores()`. You will require the parallel package for this.

Then you can do the following:

At the start of every R session, you can set the number of cores accessed during OpenMP parallel execution by issuing the command `options(rf.cores = x)`, where `x` is the number of cores. If `x` is a negative number, the package will access the maximum number of cores on your machine. The options command can also be placed in the users .Rprofile file for convenience. You can, alternatively, initialize the environment variable `RF_CORES` in your shell environment.

The default value for `rf.cores` is -1 (-1L), if left unspecified, which uses all available cores, with a minimum of two.

R-side Parallel Processing – Setting the Number of CPUs

The package also implements R-side parallel processing via the `parallel` package contained in the base `R` distribution. However, the `parallel` package must be explicitly loaded to take advantage of this functionality. When this is the case, the R function `lapply()` is replaced with the parallel version `mclapply()`. You can set the number of cores accessed by `mclapply()` by issuing the command

```
options(mc.cores = x)
```

where `x` is the number of cores. The options command can also be placed in the users .Rprofile file for convenience. You can, alternatively, initialize the environment variable `MC_CORES` in your shell environment. See the help files in parallel for more information.

The default value for `mclapply()` on non-Windows systems is two (2L) cores. On Windows systems, the default value is one (1L) core.

Example: Setting the Number of CPUs

As an example, issuing the following options command uses all available cores for both OpenMP and R-side processing:

```
options(rf.cores=detectCores(), mc.cores=detectCores())
```

As stated above, this option command can be placed in the users .Rprofile file.

Cautionary Note on Parallel Execution

1. Once the package has been compiled with OpenMP enabled, trees will be grown in parallel using the rf.cores option. Independently of this, we also utilize mclapply() to parallelize loops in R-side pre-processing and post-processing of the forest. This is always available and independent of whether the user chooses to compile the package with the OpenMP option enabled. 2. It is important to NOT write programs that fork R processes containing OpenMP threads. That is, one should not use mclapply() around the functions rfsrc(), predict.rfsrc(), vimp.rfsc(), var.select.rfsrc(), find.interaction.rfsrc() and partial.rfsrc(). In such a scenario, program execution is not guaranteed.

Cite this vignette as

H. Ishwaran, M. Lu, and U. B. Kogalur. 2021. "randomForestSRC: parallel processing vignette." http://randomforestsrc.org/articles/parallel.html.

```
@misc{HemantParallel,
    author = "Hemant Ishwaran and Min Lu and Udaya B. Kogalur",
    title = {{randomForestSRC}: parallel processing vignette},
    year = {2021},
    url = {http://randomforestsrc.org/articles/parallel.html}
}
```