Introduction to R and Statlab Server

Xuelong Wang

February 28, 2019

How to connect to the Statlab Server

- Fixed IP address 131.193.178.77
- hostname: statlab.math.uic.edu
- Mac users
  - Terminal.app
  - ssh `username@statlab.math.uic.edu`
- Windows users
  - PuTTY
  - open source SSH client for windows
  - `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

How to use the R.Studio

- R
  - Using Terminal or PuTTY
  - just type R
- Rstudio
  - Web browser e.g. Chrome
  - `http://statlab.math.uic.edu:8787/`
- some notes about using Rstudio
  - save all the documents
  - save and delete all objects in the Global Enviroment (rm(list=ls()))
  - using top and kill the terminate your R sessions

foreach package

An easy and starndard way of parallel comuptation

- Can run a for-loop task as a set of of parallel tasks
- Take care of the communication between the tasks (cores)

Getting start Example

Calculate the sum of the square

$$\sum_{i=1}^{10000} \sum_{j=1}^{i} j^2$$

There is a warning saying the loop ran sequentially
To run the loop parallelly, we need to register parallel backends.

```
system.time(foreach(i = 1:10000) %do% sum((1:i)^2))[3]
## elapsed
##   3.228
system.time(foreach(i = 1:10000) %dopar% sum((1:i)^2))[3]
## Warning: executing %dopar% sequentially:
## no parallel backend registered
## elapsed
##   3.116
```

Parallel backends

```
registerDoParallel(cores = 2)
getDoParWorkers()
## [1] 2
```

registerDoParallel() is used to register cores to parallel computation

```
system.time(foreach(i = 1:10000) %do% sum(sqrt(1:i)))[3]
## elapsed
##   3.316
system.time(foreach(i = 1:10000) %dopar% sum(sqrt(1:i)))[3]
## elapsed
##   2.804
```

Bootstraping example

```
dim(x)
## [1] 100   2
r <- foreach(i = 1:10000) %dopar% {
    ind <- sample(100, 100, replace = TRUE)
    result1 <- glm(x[ind, 2] ~ x[ind, 1], family = binomial
    coefficients(result1)
}
```

- Escaped time for using 2 cores is 14.055 seconds
- Escaped time for using single core is 26.611 seconds

data.table

Why data.table

- Data.table is an extension of data.frame package in R
- Much Faster than other structures in R
- SQL type syntax: `DT[where, select|update|do, by]`

How to create a data.table

data.table function

```
library(data.table)
DT = data.table(x = c("b", "b", "b", "a",
    "a"), v = rnorm(5))
CARS = data.table(cars)
tables()
##     NAME NROW NCOL MB       COLS KEY
## 1: CARS   50    2  0 speed,dist
## 2:   DT    5    2  0        x,v
## Total: 0MB
```

How to create a data.table

Subset

```
library(data.table)
DT[2, ]
##    x          v
## 1: b -0.5867648
DT[x == "a", ]
##    x          v
## 1: a -0.2935104
## 2: a -0.5303302
cat(try(DT["a", ], silent = TRUE))
## Error in `[.data.table`(DT, "a", ) :
##   When i is a data.table (or character vector), the colu
```

How to create a data.table

setKeys

```
setkey(DT, x)
DT["a", ]
##    x           v
## 1: a -0.2935104
## 2: a -0.5303302
```

- Note that the keys are not requried to be unique.

How fast

data.frame

```
grpsize = ceiling(1e+07/26^2)  # 10 million rows, 676 grou
DF <- data.frame(x = rep(LETTERS, each = 26 *
    grpsize), y = rep(letters, each = grpsize),
    v = runif(grpsize * 26^2), stringsAsFactors = FALSE)
head(DF, 3)
##   x y         v
## 1 A a 0.5956366
## 2 A a 0.6750965
## 3 A a 0.4565744
dim(DF)
## [1] 10000068        3
```

How fast

data.frame

```
tt = system.time(ans1 <- DF[DF$x == "R" &
    DF$y == "h", ])  # 'vector scan'
tt
##    user  system elapsed
##   0.188   0.044   0.233
head(ans1, 3)
##         x y         v
## 6642058 R h 0.1720161
## 6642059 R h 0.4975197
## 6642060 R h 0.2276249
```

How fast

data.frame

```
DT = as.data.table(DF)
system.time(setkey(DT, x, y))
##    user  system elapsed
##   0.128   0.016   0.143
ss = system.time(ans2 <- DT[list("R", "h")])
ss
##    user  system elapsed
##   0.000   0.000   0.003
head(ans2, 3)
##    x y         v
## 1: R h 0.1720161
## 2: R h 0.4975197
## 3: R h 0.2276249
```