Stat 341, Homework 1

Jiansong Xu

2019-01-25

1. (1 mark) The following simulation function simulates n replicates of an explanatory variable X and a response variable Y = βX + E, where β is a regression coefficient between −1 and 1 and E ∼ N(0, 1) is random noise. Run the code chunk and then use the function to simulate one dataset of size n = 1000 and save the result in an object called `dd`.

```
simdat <- function(n) {
  beta <- runif(1,min=-1,max=1)
  x <- rnorm(n)
  y <- beta * x + rnorm(n)
  data.frame(x=x,y=y)
}
```

```
dd <- simdat(1000)
```

2. (2 marks) Create a larger dataset by calling `simdat()` `N=500` times over and stacking the results. The larger dataset should have `500*1000` rows and 2 columns. Call your stacked dataset `bigd1`. To create the stacked dataset, initialize with `bigd1 <- NULL` and use a `for` loop to build up `bigd1` one layer at a time. Time this code using the `system.time()` function. An example use of `system.time()` to time an R command, e.g., `x <- rnorm(100000)` is:

```
system.time({
  x <- rnorm(100000) # Could put multiple lines of R code here
})
```

```
##    user  system elapsed
##       0       0       0
```

Use the first element of the output (`user` time) as your measure of execution time.

Answer

```
system.time({
N <- 500
bigd1 <- NULL
for (i in 1:N) {
  bigd1 <- rbind(bigd1, simdat(1000))
}})
```

```
##    user  system elapsed
##    6.05    2.29    8.36
```

3. (2 marks)

Repeat 2, but this time, instead of stacking the output of `simdat()`, coerce the output of `simdat()` to a matrix, and stack the matrices. Use `system.time()` to time your code and compare the timing from question (2).

Answer

```
system.time({
bigd1.1 <- NULL
for (i in 1:N) {
  bigd1.1 <- rbind(bigd1.1, data.matrix(simdat(1000)))
}})
```

```
##    user  system elapsed
##    1.18    0.81    2.00
```

Output as a matrix takes less system time to run.

4. (3 marks) Now build `bigd2` by (i) initializing an empty matrix of appropriate dimension, and (ii) looping 500 times and inserting simulated datasets of size n = 1000, coerced to matrices, into successive layers of `bigd2`. Time this code and compare the timing to that of part (3). You may find the following R function useful:

```
layerInds <- function(layerNum,nrow) {
  ((layerNum-1)*nrow + 1):(layerNum*nrow)
}
# Example use:
inds <- layerInds(layer=1,nrow=1000)
range(inds)
```

```
## [1]    1 1000
```

Answer

```
system.time({
bigd2 <- matrix(nrow = 500000, ncol = 2)
for ( i in 1:N){
  inds <- layerInds(layer = i, nrow = 1000)
  bigd2[inds,] <- data.matrix(simdat(1000))
}
})
```

```
##    user  system elapsed
##    0.44    0.01    0.45
```

This ... ss time than part 3.