STAT 33B Workbook 11

Ming Fong (3035619833)

Nov 12, 2020

This workbook is due **Nov 12, 2020** by 11:59pm PT.

The workbook is organized into sections that correspond to the lecture videos for the week. Watch a video, then do the corresponding exercises *before* moving on to the next video.

Workbooks are graded for completeness, so as long as you make a clear effort to solve each problem, you'll get full credit. That said, make sure you understand the concepts here, because they're likely to reappear in homeworks, quizzes, and later lectures.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

In the notebook, you can run the line of code where the cursor is by pressing `Ctrl` + `Enter` on Windows or `Cmd` + `Enter` on Mac OS X. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

Code Performance

Watch the "Code Performance" lecture video.

No exercises for this section.

Profiling

Watch the "Profiling" lecture video.

No exercises for this section.

Profvis

Watch the "Profvis" lecture video.

Exercise 1

Read chapter 23 of Advanced R.

The flame graph for the function `rhand` function in the lecture has entries for a call to `<GC>`. What do these entries mean? Why are the listed as being "called" by the `c` function? Explain in 3-5 sentences.

YOUR ANSWER GOES HERE:

Calls to `<GC>` are for the garbage collector, which clears unused memory. The `c` function calls this because of its copy-on-write. Everytime you replace the value of c, a new copy is created and the old one needs to be removed to free up memory. This is inefficient and should be avoided if possible.

Profiling Case Study

Watch the "Profiling Case Study" lecture video.

The next two exercises are related to the "Code Performance" lecture, but I've put them here at the end so the assigned readings are in order.

Exercise 2

Read chapter 24 of Advanced R.

What's the difference between `colSums` and `.colSums`? Explain.

YOUR ANSWER GOES HERE:

`.colSums` can only be used with numeric matrices and is the bare-bones version of `colSums`. `.colSums` requires specifying the number of rows and columns in the matrix.

Exercise 3

Create several test matrices of different sizes and use the microbenchmark or bench package to compare the speeds of:

- `colSums(m)`
- `.colSums(m)`
- `apply(m, 2, sum)`

Here `m` denotes a matrix. Use a plot to present your results. Is one of these consistently faster than the others? Does the size of the matrix have an effect on which is faster?

YOUR ANSWER GOES HERE:

```
library(microbenchmark)
n = 1000
mats = list()
for(x in 1:n) {
    mats[[x]] = matrix(1:(x^2), nrow = x, ncol = x)
}

times = data.frame(
    n = rep(NA, n),
    colSums = rep(NA, n),
    .colSums = rep(NA, n),
    apply = rep(NA, n)
)
for(x in 1:n) {
```

```
    times[x, "n"] = x
    times[x, "colSums"] = summary(microbenchmark(colSums(mats[[x]]), unit = "us"))[["mean"]]
    times[x, ".colSums"] = summary(microbenchmark(.colSums(mats[[x]], x, x), unit = "us"))[["mean"]]
    times[x, "apply"] = summary(microbenchmark(apply(mats[[x]], 2, sum), unit = "us"))[["mean"]]
}
```

It seems that `.colSums` is the fastest method of the three. The size of the matrix has little effect on the order of speeds (`colSums` and `.colSums` occasionally switch). `apply` is definitely slower than the other two.

```
library(ggplot2)
ggplot(times, aes(x = n)) +
    geom_line(aes(y = colSums, color = "colSums")) +
    geom_line(aes(y = .colSums, color = ".colSums")) +
    geom_line(aes(y = apply, color = "apply")) +
    labs(title = "Time vs n", color = "Function") +
    xlab("n") +
    ylab("Time (us)") +
    theme()
```