STAT 33B Lecture – April 22

Topics:

- Profiling
- Improving Performance

Profiling

Profiling means analyzing code *as it runs* to determine:

- Time spent in each function
- Number of calls to each function
- Memory usage

A statistical profiler, like R’s, periodically checks which function is running.

The **profvis** package provides additional visualization tools.

Making Code Faster, Part 1

Things to try:

1. Look for packages
2. Use specialized functions (rather than general-purpose)
3. Do less
4. Vectorize

The Space-time Trade-off

- Making code faster may make it use more space (memory or storage).
- Making code use less space may make it slower.

This trade-off arises in many programming problems.

Examples:

- Memoization
- Vectorization versus looping (in some cases)

Making Code Faster, Part 2

Run code in parallel:

- Built-in **parallel** package for simple parallelism.
- Many problems are difficult to parallelize.

Call C, C++, or Fortran code:

- Higher potential for unpleasant bugs.
- See Wickham, Ch. 19 & 20.