Lecture 13 – Profiling and Optimizing code in R

Learning Objectives:

3. Learn the basic principles of software design.

    3.4. Learn about optimization and profiling code.

What is ‘Optimization’?

- any modifications that you make to code that improves quality, efficiency, and/or speed

- Can have a variety of goals: execution speed, memory usage, minimizing inputs, making the program smaller

- We’ll cover two: speed (today) and memory (Monday)

- Often these work hand-in-hand!

- Tradeoffs exist between speed and efficiency or flexibility. Thinking carefully about what ***you want out of optimization*** will reduce the amount of time you spend optimizing code.

Should you optimize?

- Should you make it faster? Is it worth the time to optimize it?

- How much faster does it need to be?

- How much time should you devote to making it faster?

Why is R so slow?

- *Extreme dynamism* (very dynamic and flexible, gives up speed)

- *Name lookup*: lexical scoping during function calling (like, everything is a function)

- *Lazy evaluation*: promise objects creates extra stuff that slows everything down

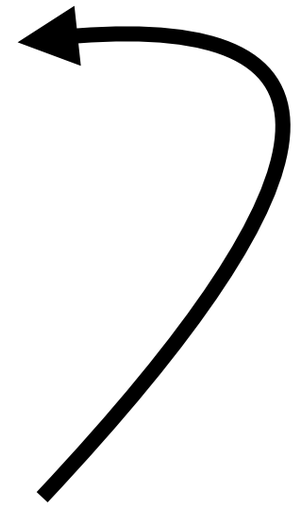
… but mostly because code is not efficient.

The Optimization Process:

1. Find the biggest time suck.

2. Try to fix that one (might not work).

3. Start over until you're satisfied.



How do you find this?

Profile!

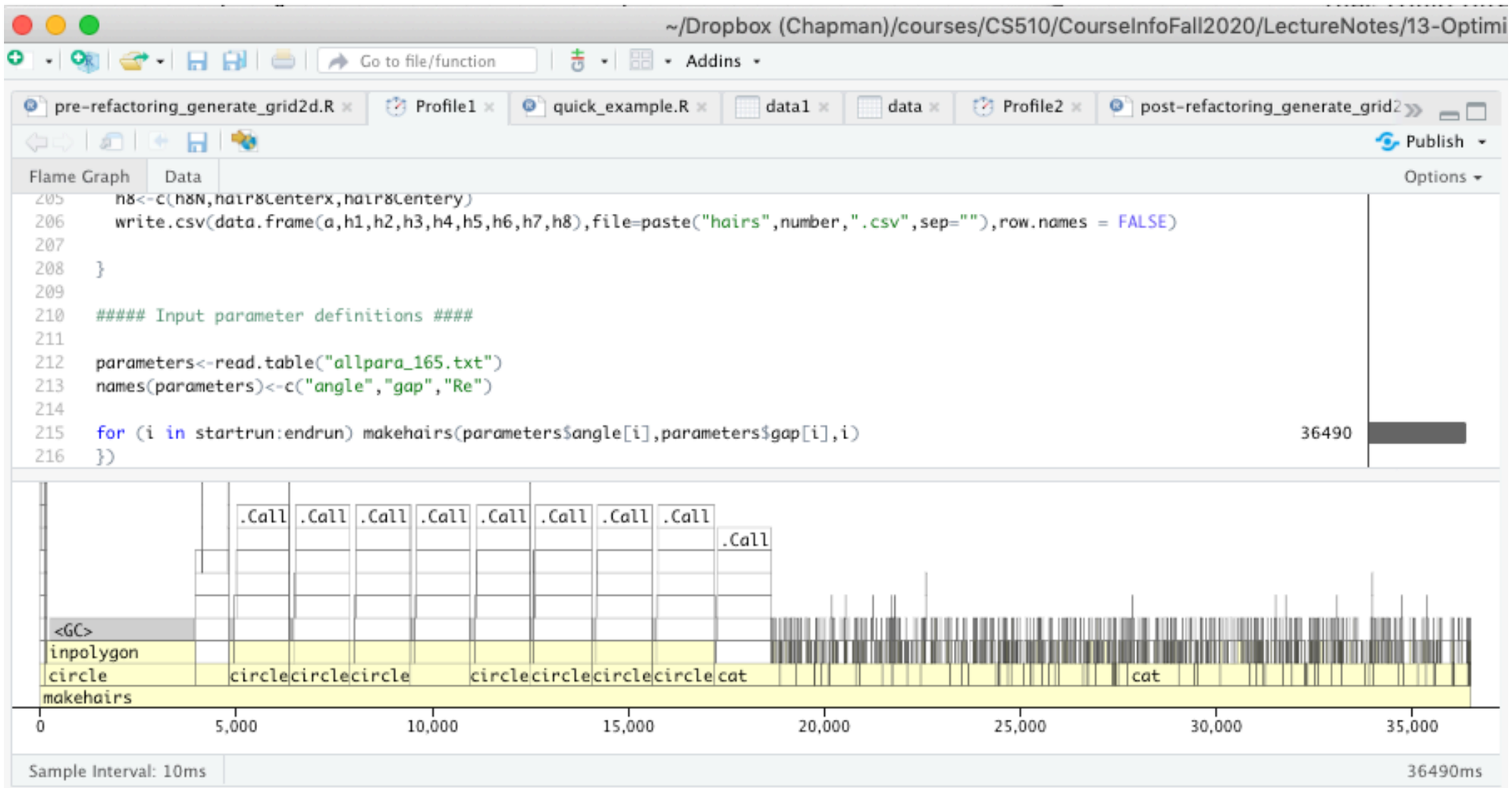
Try not to rely on your instincts.

Just Profile It

Profilers in R

`profvis` package: RStudio's built-in profiler.

ProfilingExample in 13-OptimizinginR



`microbenchmark` package: high-precision profiler for R.

```
> x = runif(100)
> microbenchmark(sqrt(x), x^0.5)
```

Common and Quick Strategies for Improving Speed

General advice:

- Keep a record
- Generate a representative test case
- Set a target speed / time amount

Specific strategies:

- Look for existing solutions
- Do less work with more appropriate functions
- Vectorize!
- Parallelize (in a few weeks), avoid copies (next time)
- Byte-code compile
- Rewrite in something faster

More Information

https://github.com/jennybc/code-smells-and-feels – "Code Smells and Feels" by Jenny Bryan, a talk at the UseR conference 2018

http://silab.fon.bg.ac.rs/wp-content/uploads/2016/10/Refactoring-Improving-the-Design-of-Existing-Code-Addison-Wesley-Professional-1999.pdf – Refactoring: Improve the Design of Existing Code by Martin Fowler