R Optimization

VP Nagraj

optimization

noun op·ti·mi·za·tion \ ˌäp-tə-mə-ˈzā-shən \

an act, process, or methodology of making something (such as a design, system, or decision) as fully perfect, functional, or effective as possible; specifically : the mathematical procedures (such as finding the maximum of a function) involved in this

https://www.merriam-webster.com/dictionary/optimization

R is slow ...

language (R)

- extreme dynamism
- lexical scoping
- lazy evaluation

implementation (GNU-R)

- in-place conditions vs `ifelse()`
- extracting single a value from a data frame
- `for` loops

http://adv-r.had.co.nz/Performance.html#why-is-r-slow

> Beyond performance limitations due to design and implementation, it has to be said that a lot of R code is slow simply because it's poorly written. Few R users have any formal training in programming or software development. Fewer still write R code for a living. Most people use R to understand data: it's more important to get an answer quickly than to develop a system that will work in a wide variety of situations.

takeaway: yes in some regards R is inherently slow(er than other languages) ... but there are still plenty of ways to potentially make *your* code faster

slow? how do you know?

```r
distro <- function(size = 1e7, type = "normal", ...) {

  x <-
    switch(type,
           "normal" = rnorm(size, ...),
           "uniform" = runif(size, ...),
           "poisson" = rpois(size, ...)
           )

  hist(x)

}
```

base benchmarking

```
# capture time before evaluation
st <- Sys.time()

# evaluate ...
distro()

# capture time after evaluation
et <- Sys.time()

# difference?
et - st
```

```
# wrap evaluation in system.time
system.time({

  distro()

})
```

mirobenchmark()

```
library(microbenchmark)
microbenchmark(distro(), times = 10)
```

```
microbenchmark(
  normal = distro(size = 1e5, type = "normal"),
  unif = distro(size = 1e5, type = "uniform"),
  pois = distro(size = 1e5, type = "poisson", lambda = 2)
  )
```

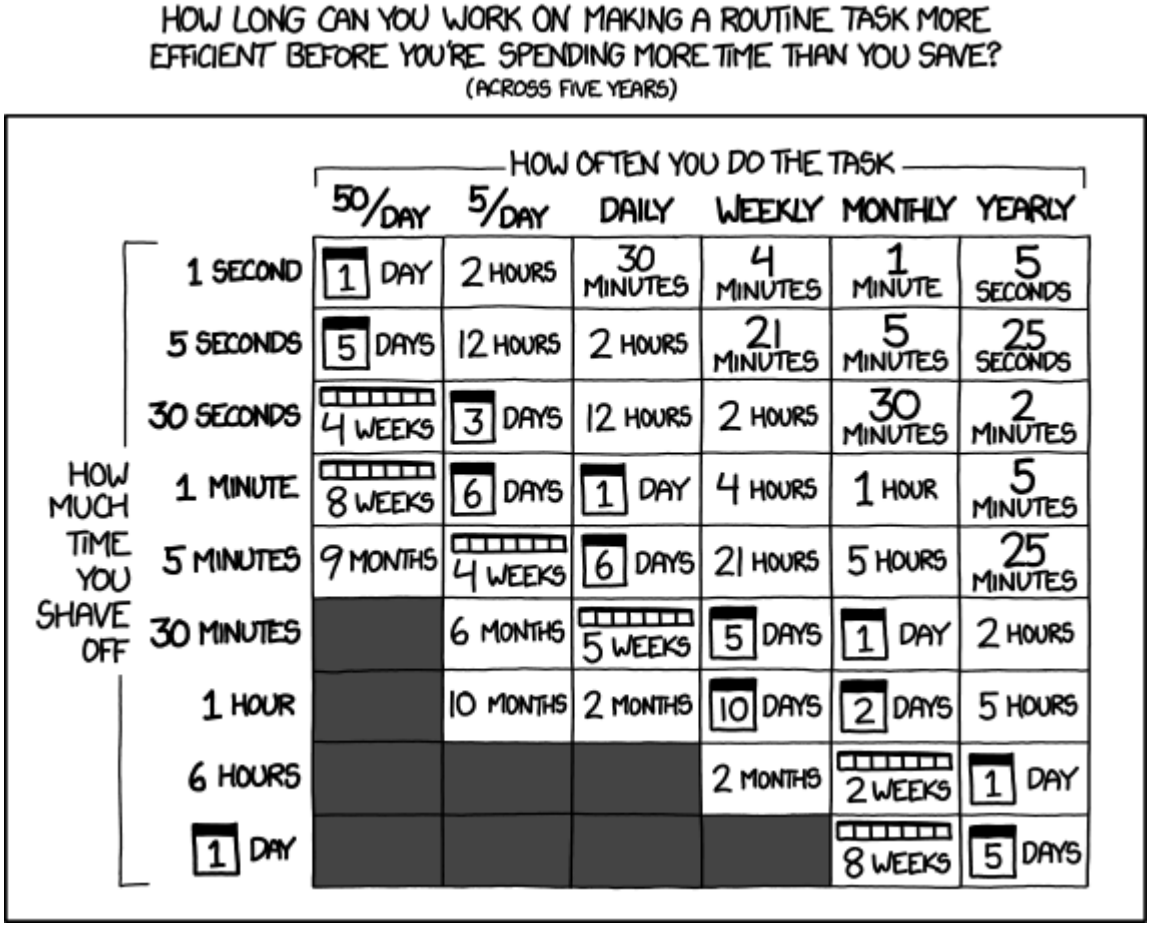
profvis()

```
library(profvis)
profvis(distro(size = 1e7, type = "uniform"))
```

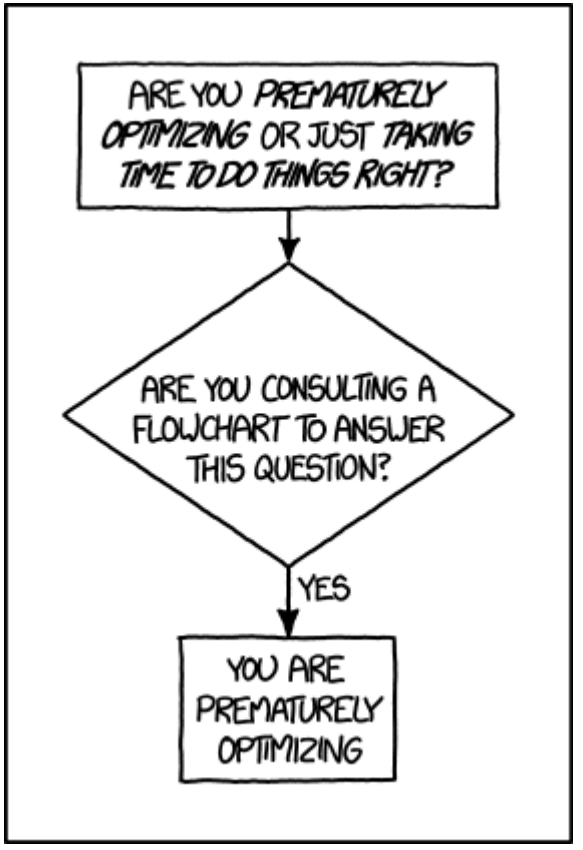
keep in mind ...

- is it worth it? (optimization energy)
- slow code > broken code (`all.equal`)
- *absolute* versus *relative* measures of speed (wall clock)

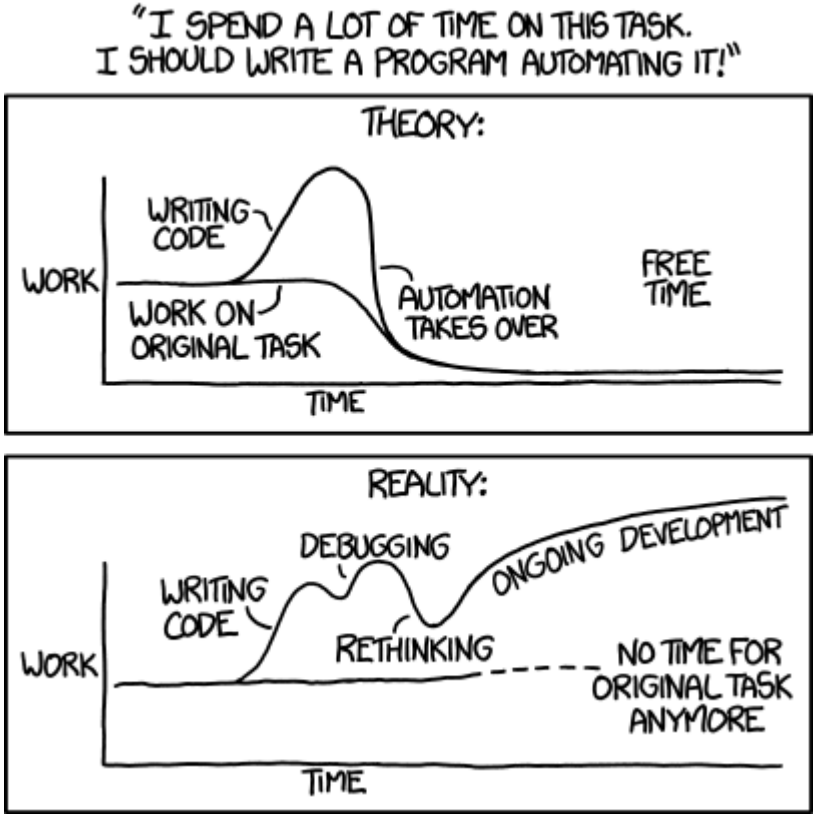
optimization energy: https://xkcd.com/1205/



optimization energy: https://xkcd.com/1691/



optimization energy: https://xkcd.com/1319/



all.equal

```
l <- replicate(100,
               sample(1:500, size = 100),
               simplify = FALSE)

microbenchmark({

  res_loop <- vector()

  for (i in 1:length(l)) {

    tmpres <- mean(l[[i]])
    res_loop <- c(res_loop,tmpres)

  }
})

microbenchmark({

  res_lapply <- lapply(l, mean)

})

all.equal(res_loop, res_lapply)
```

wall clock

```
x <- runif(1e7)

microbenchmark(
  method1 = which(x == min(x)),
  method2 = which.min(x)
  )
```

```
## Unit: milliseconds
##     expr      min       lq     mean   median       uq       max neval
##  method1 55.35757 60.10862 69.03225 66.36669 70.27252 132.62209   100
##  method2 16.14738 17.47171 18.46979 18.78416 19.21101  22.75218   100
```

wall clock

```
microbenchmark(
  method1 = chickwts[49,2],
  method2 = chickwts$feed[49]
)
```

```
## Unit: microseconds
##     expr    min     lq     mean  median      uq      max neval
##  method1 14.961 15.515 16.37856 15.7775 16.1855   44.139   100
##  method2  9.663 10.048 38.70925 10.4600 10.7570 2826.604   100
```

credits

Advanced R Programming (Hadley Wickham)

http://adv-r.had.co.nz/Performance.html

Efficient R Programming (Colin Gillespie and Robin Lovelace)

https://csgillespie.github.io/efficientR/

Optimizing R Code workshop (Jackie Huband)