Debugging and Profiling in R

Emil Hvitfeldt

2019-2-19

Fix performance

Debugging

Measure performance

Profiling / Benchmarking

Improve performance

Code improvements

The Art of identifying the right line(s) of code

The Art of identifying the right line(s) of code

Identify bottlenecks

The Art of identifying the right line(s) of code

Identify bottlenecks

Isolate problem

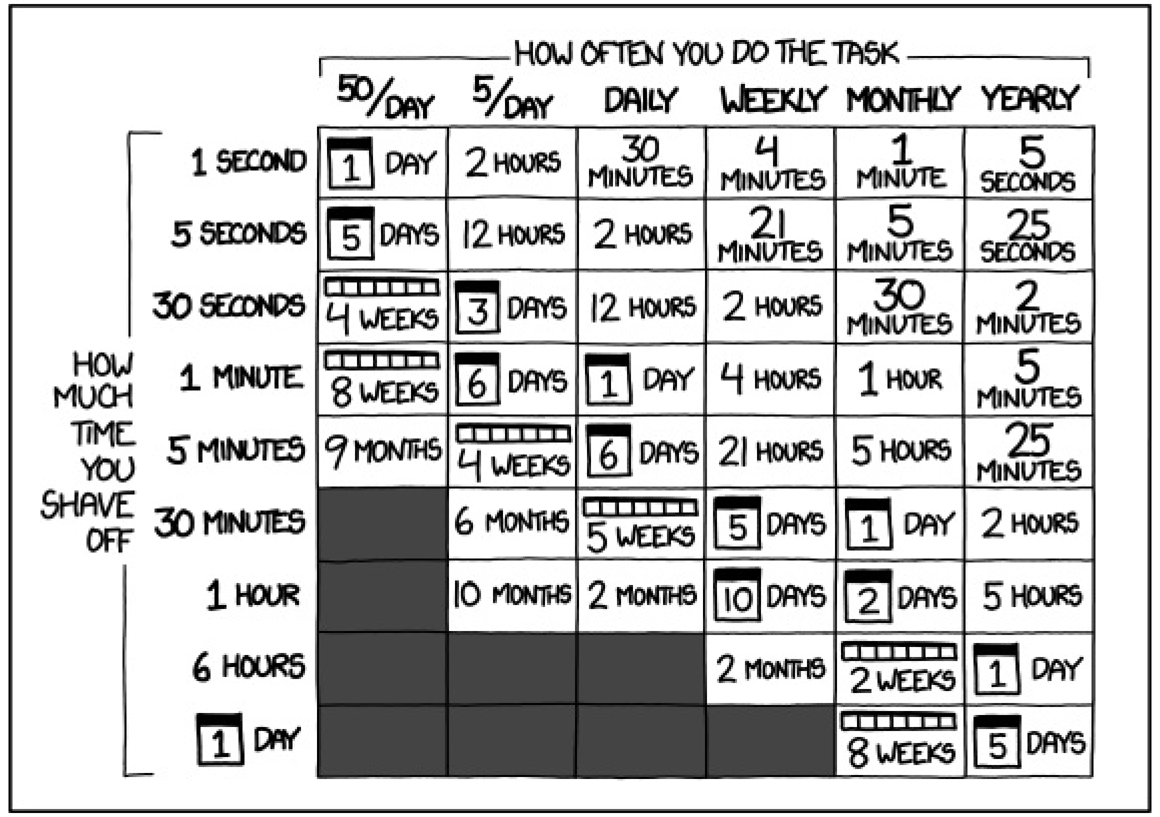
The Art of identifying the right line(s) of code

Identify bottlenecks

Isolate problem

Create reproducible example

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)



HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

Debugging

> Debugging is like being the detective in a crime movie where you're also the murderer.

Filipe Fortes

There are 2 types of errors

Getting an error

Don't get expected outcome

There are 2 many types of errors

Getting an error

Getting a warning
R crashes

Don't get expected outcome

Test failed
no outcome

Plan of attack

google the error message

Very real chance that someone elser had the same problem you just had.

Isolate the problem

Your problem will most likely be confined to one area of your code.

Make it repeatable

Work towards a minimal reproducible error.

Call/Ask a friend

It can be hard to google something if you don't know the name of the thing you want or have a hard time describing it concisely.

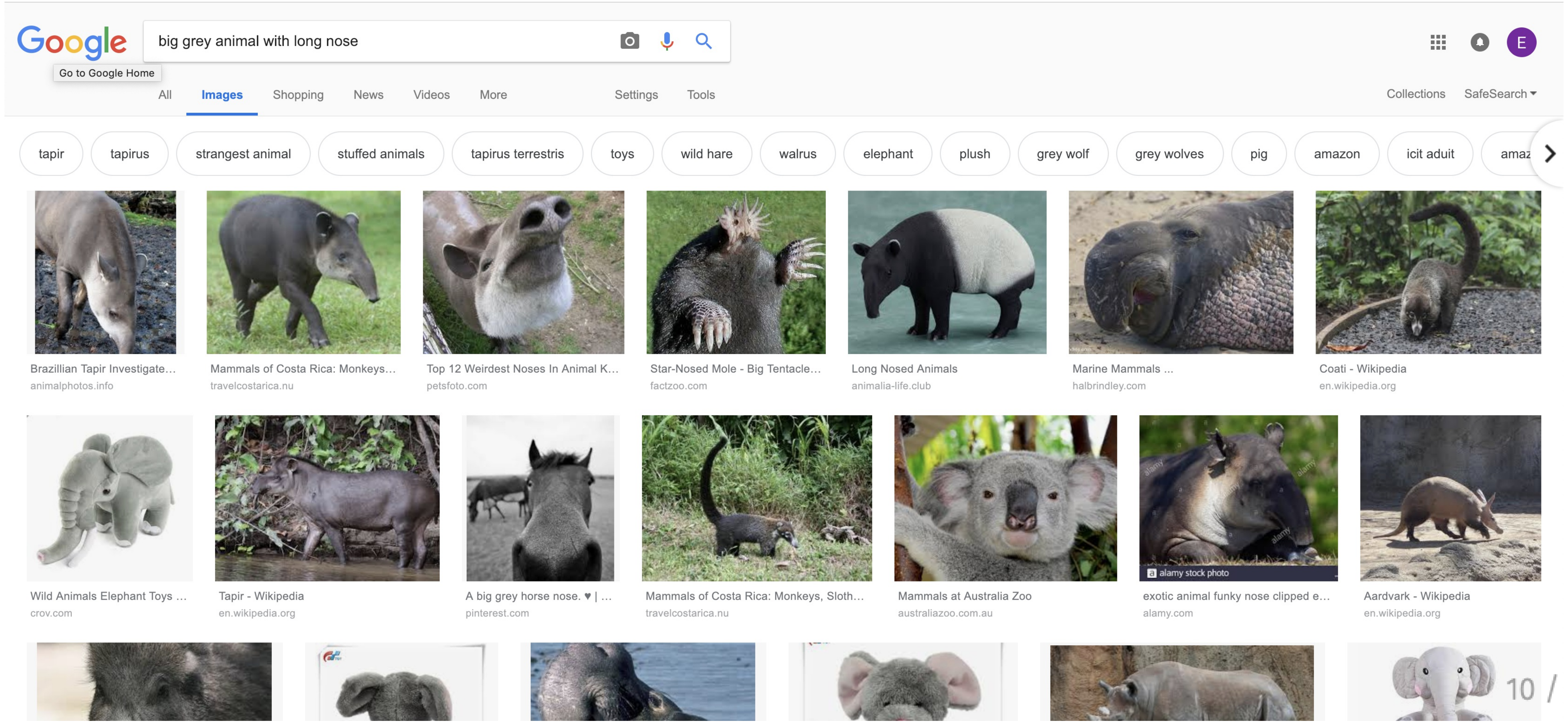
Problem

I have a list of numbers and I want to add each number to all the previous numbers in a list.

Solution

> you are thinking of a cumulative sum, implemented in R as `cumsum()`.

friend



big grey animal with long nose

Brazillian Tapir Investigate ...
animalphotos.info

Mammals of Costa Rica: Monkeys ...
travelcostarica.nu

Top 12 Weirdest Noses In Animal K...
petsfoto.com

Star-Nosed Mole - Big Tentacle ...
factzoo.com

Long Nosed Animals
animalia-life.club

Marine Mammals ...
halbrindley.com

Coati - Wikipedia
en.wikipedia.org

Wild Animals Elephant Toys ...
crov.com

Tapir - Wikipedia
en.wikipedia.org

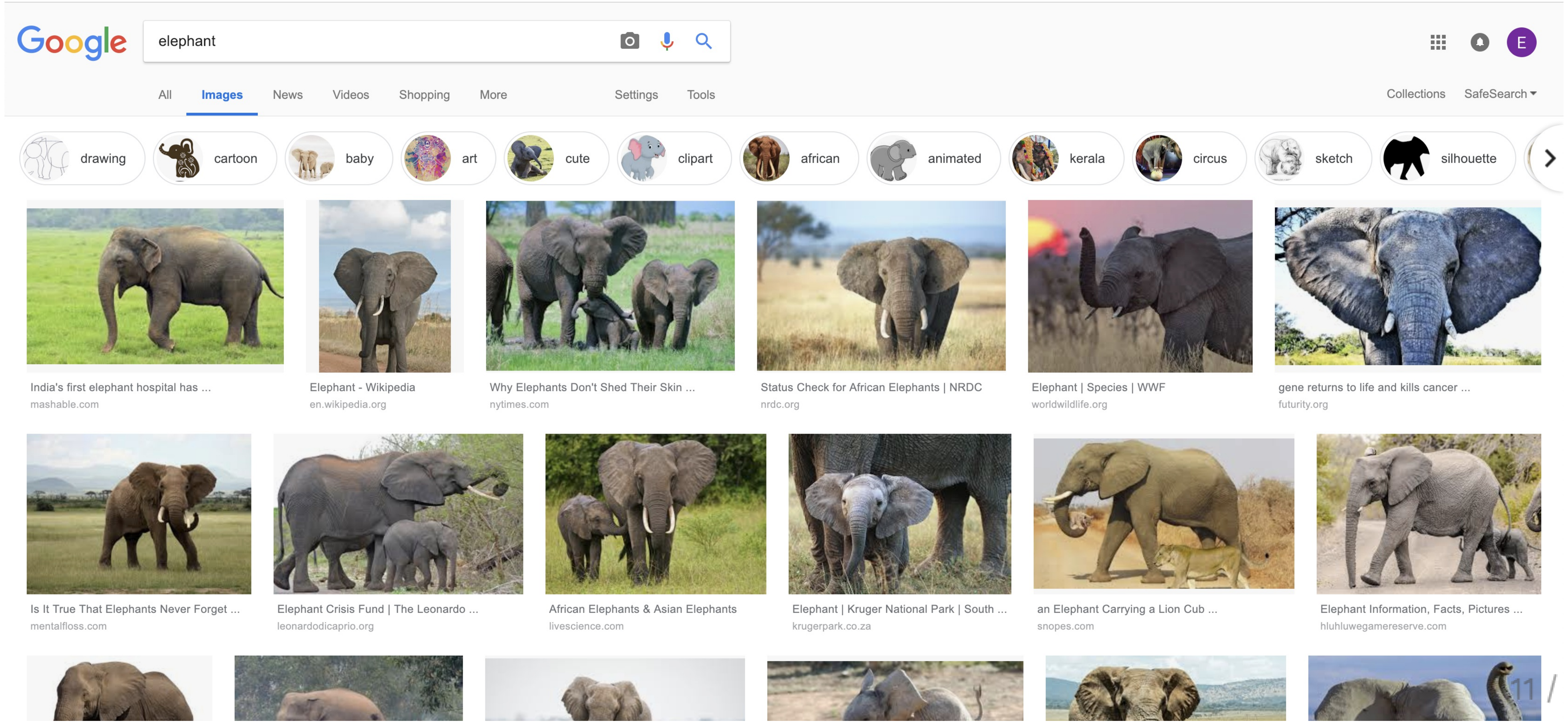
A big grey horse nose. ♥ | ...
pinterest.com

Mammals of Costa Rica: Monkeys, Sloth...
travelcostarica.nu

Mammals at Australia Zoo
australiazoo.com.au

exotic animal funky nose clipped e...
alamy.com

Aardvark - Wikipedia
en.wikipedia.org



elephant

India's first elephant hospital has ...
mashable.com

Elephant - Wikipedia
en.wikipedia.org

Why Elephants Don't Shed Their Skin ...
nytimes.com

Status Check for African Elephants | NRDC
nrdc.org

Elephant | Species | WWF
worldwildlife.org

gene returns to life and kills cancer ...
futurity.org

Is It True That Elephants Never Forget ...
mentalfloss.com

Elephant Crisis Fund | The Leonardo ...
leonardodicaprio.org

African Elephants & Asian Elephants
livescience.com

Elephant | Kruger National Park | South ...
krugerpark.co.za

an Elephant Carrying a Lion Cub ...
snopes.com

Elephant Information, Facts, Pictures ...
hluhluwegamereserve.com

Hunting tools

traceback()

debug()

breakpoints

broswer()

Urn simulation

```r
main_function <- function(n_max, n_black, balls, n) {
  check_input(n_max, n_black, balls, n)
  x_prep <- prep_data(n_black, balls)

  res <- numeric(n)
  for(i in seq_len(n)) {
    data <- simulate_data(x_prep, n_max)
    res[i] <- analyse_results(data)
  }
  res
}
```

```r
check_input <- function(n_max, n_black, balls, n) {
  if(!is.numeric(n_max))
    stop("`n_max` must be numeric.")
  if(!is.numeric(n_black))
    stop("`n_black` must be numeric.")
  if(!is.numeric(balls))
    stop("`balls` must be a numeric.")
  if(!is.numeric(n))
    stop("`n` must be a numeric.")

    if(length(n_max) != 1)
    stop("`n_max` must have length 1.")
  if(!is.numeric(n_black))
    stop("`n_black` must have length 1.")
  if(!is.numeric(n))
    stop("`n` must have length 1.")
}
```

```r
prep_data <- function(n_black, balls) {
  c(rep(0, n_black), ball_create(balls))
}

ball_create <- function(balls) {
  ball_id <- seq_len(balls)
  res <- numeric()
  for(i in ball_id) {
    res <- c(res, rep(ball_id[i], balls[i]))
  }
  res
}
```

```r
simulate_data <- function(urn, n_max) {
  for (j in length(urn):n_max) {
    draw <- sample(urn, 1)
    if(draw == 0) {
      urn <- c(urn, max(urn) + 1)
    } else {
      urn <- c(urn, draw)
    }
  }
  urn
}
```

```r
analyse_results <- function(x) sum(x == 1)
```

```r
options(warn = 2)
main_function(n_max = 50, n_black = 1, balls = c(1, 1), n = 100)
traceback()
```

```
## 7: doWithOneRestart(return(expr), restart)
## 6: withOneRestart(expr, restarts[[1L]])
## 5: withRestarts({
##        .Internal(.signalCondition(simpleWarning(msg, call), msg,
##            call))
##        .Internal(.dfltWarn(msg, call))
##    }, muffleWarning = function() NULL)
## 4: .signalSimpleWarning("first element used of `length.out` argument",
##        quote(seq_len(balls))) at #2
## 3: ball_create(balls) at #2
## 2: prep_data(n_black, balls) at #3
## 1: main_function(n_max = 50, n_black = 1, balls = c(1, 1), n = 100)
```

Using browser() and breakpoints

Live Demo

urn_code.R

debug() and debugonce()

```
debug(ball_create)
main_function(n_max = 50, n_black = 1, balls = c(1, 1), n = 100)

debugonce(simulate_data)
main_function(n_max = 50, n_black = 1, balls = c(1, 1), n = 100)
```

Write tests for your code

For every fixed bug

Benchmarking

> Don't fix something that is running fast enough.
>
> Unknown

2 types of benchmarking

Slow (time > 1 sec)

`system.time()`

tictoc package

Fast (time < 1 sec)

Microbenchmarking
bench package

Timing slow code

```r
fibonacci <- function(n) {
  if(n == 0) {
    return(0)
  }
  if(n == 1) {
    return(1)
  }
  fibonacci(n - 1) + fibonacci(n - 2)
}
```

Timing slow code

```
fibonacci <- function(n) {
  if(n == 0) {
    return(0)
  }
  if(n == 1) {
    return(1)
  }
  fibonacci(n - 1) + fibonacci(n - 2)
}
```

```
system.time(
  fibonacci(30)
)
```

```
##    user  system elapsed 
##   0.850   0.001   0.851 
```

Timing slow code

```
fibonacci <- function(n) {
  if(n == 0) {
    return(0)
  }
  if(n == 1) {
    return(1)
  }
  fibonacci(n - 1) + fibonacci(n - 2)
}
```

```
system.time(
  fibonacci(1)
)
```

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

tictoc package for timing

```r
library(tictoc)

tic()
X <- fibonacci(5)
toc()
```

```
## 0.005 sec elapsed
```

```r
tic("fibonacci with n = 5")
X <- fibonacci(5)
toc()
```

```
## fibonacci with n = 5: 0.002 sec elapsed
```

tictoc package for timing

```r
library(tictoc)

tic("Total")
  tic("n = 4")
  X <- fibonacci(4)
  toc()

  tic("n = 5")
  X <- fibonacci(5)
  toc()

  tic("n = 6")
  X <- fibonacci(6)
  toc()
toc()
```

```
## n = 4: 0.002 sec elapsed
## n = 5: 0.002 sec elapsed
## n = 6: 0.001 sec elapsed
## Total: 0.008 sec elapsed
```

Microbenchmarking with bench package

Live Demo

Notice the units

- 1 ms, then one thousand calls takes a second.
- 1 µs, then one million calls takes a second.
- 1 ns, then one billion calls takes a second.

Profiling

> Never mess with someone who has more spare time than you do[.]

Fredrik Backman, My Grandmother Asked Me to Tell You She's Sorry

Live Demo

urn_profile.R

Profiler information

R uses a sampling/statistical profiler

Memory

left - allocated
right - freed

`<GC>` Garbage collection

Indication lots of small objects are being created

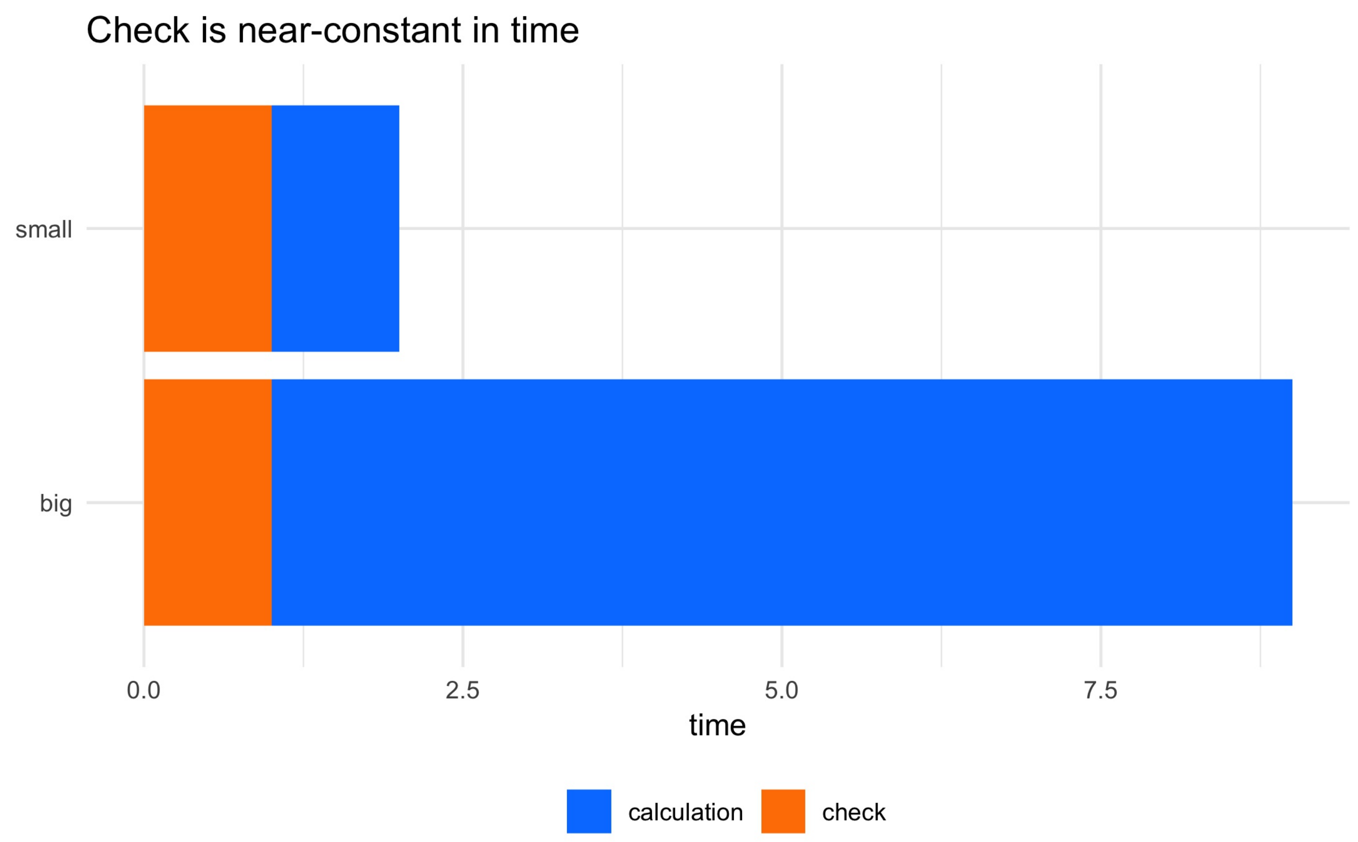
```r
x <- numeric(50000)
for(i in seq_len(50000)) {
  x <- c(x, i)
}
```

R uses copy-on-modify

flexibility and functionality > speed

```
var
```

```
## function (x, y = NULL, na.rm = FALSE, use)
## {
##     if (missing(use))
##         use <- if (na.rm)
##             "na.or.complete"
##         else "everything"
##     na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
##         "everything", "na.or.complete"))
##     if (is.na(na.method))
##         stop("invalid 'use' argument")
##     if (is.data.frame(x))
##         x <- as.matrix(x)
##     else stopifnot(is.atomic(x))
##     if (is.data.frame(y))
##         y <- as.matrix(y)
##     else stopifnot(is.atomic(y))
##     .Call(C_cov, x, y, na.method, FALSE)
## }
## <bytecode: 0x7f7f84616ea0>
## <environment: namespace:stats>
```


Check is near-constant in time
small
big
0.0
2.5
5.0
7.5
time
calculation
check

Code improvements

> “ The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time. ”

Tom Cargill

4 ways to speed up code

Buy a bigger computer

Optimize R code

Parallelize

Rewrite code in c++

4 ways to speed up code

Buy a bigger computer

Optimize R code

Parallelize

Rewrite code in c++

Pattern recogniction & trial and error

Gain speed by doing less

More examples at https://github.com/USCbiostats/software-dev/tree/master/Slow_patterns

unlist()

```
list_obj <- list(a = 1, b = 2, c = 3)

bench::mark(check = FALSE,
  unlist(list_obj),
  unlist(list_obj, use.names = FALSE)
)[c("expression", "min", "mean", "max", "itr/sec")]
```

```
## # A tibble: 2 x 5
##   expression                               min      mean       max `itr/sec`
##   <chr>                               <bch:tm> <bch:tm> <bch:tm>     <dbl>
## 1 unlist(list_obj)                       619ns     685ns    22.9µs  1459071.
## 2 unlist(list_obj, use.names = FALSE)    463ns     558ns    28.3µs  1791728.
```

table vs tabulate

```
x <- sample(x = 1:6, size = 100, replace = TRUE)
```

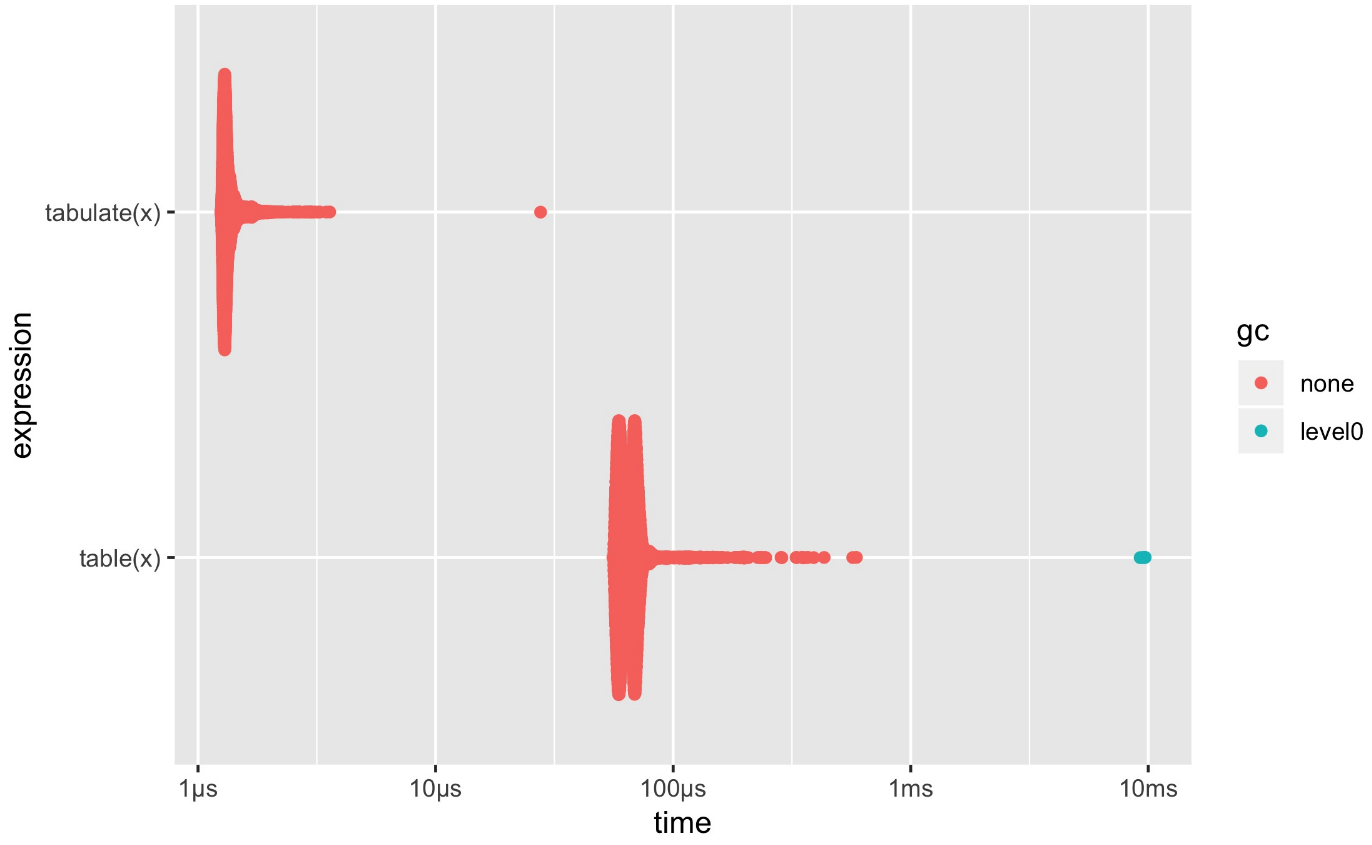
```
table(x)
```

```
## x
##  1  2  3  4  5  6 
## 13 15 16 17 17 22
```

```
tabulate(x)
```

```
## [1] 13 15 16 17 17 22
```

```
bench::mark(check = FALSE,
  table(x),
  tabulate(x)
) %>% plot()
```



Use matrix algebra

Calculate the magnitude of each point `sqrt(x^2 + y^2)`

```
x <- matrix(rnorm(20), ncol = 2)
colnames(x) <- c("x", "y")
```

```
x
```

```
##                x           y
##  [1,]  2.3991606 -0.14660420
##  [2,]  0.6249790 -0.34205853
##  [3,]  0.7698679  0.45751096
##  [4,]  0.7754097  0.04068578
##  [5,]  0.7782949 -0.10098925
##  [6,] -0.8197612  2.28329483
##  [7,]  0.4573175 -0.82314245
##  [8,]  0.8881661 -1.04656812
##  [9,] -0.1437705 -0.92939910
## [10,] -0.1813523 -0.38265007
```

Use matrix algebra

```r
x[, 1, drop = FALSE] + x[, 2, drop = FALSE]
```

```
##                x
##  [1,]  2.2525564
##  [2,]  0.2829204
##  [3,]  1.2273789
##  [4,]  0.8160955
##  [5,]  0.6773056
##  [6,]  1.4635336
##  [7,] -0.3658250
##  [8,] -0.1584020
##  [9,] -1.0731696
## [10,] -0.5640023
```

```r
y <- matrix(c(1, 1), ncol = 1)
x %*% y
```

```
##             [,1]
##  [1,]  2.2525564
##  [2,]  0.2829204
##  [3,]  1.2273789
##  [4,]  0.8160955
##  [5,]  0.6773056
##  [6,]  1.4635336
##  [7,] -0.3658250
##  [8,] -0.1584020
##  [9,] -1.0731696
## [10,] -0.5640023
```

Use matrix algebra

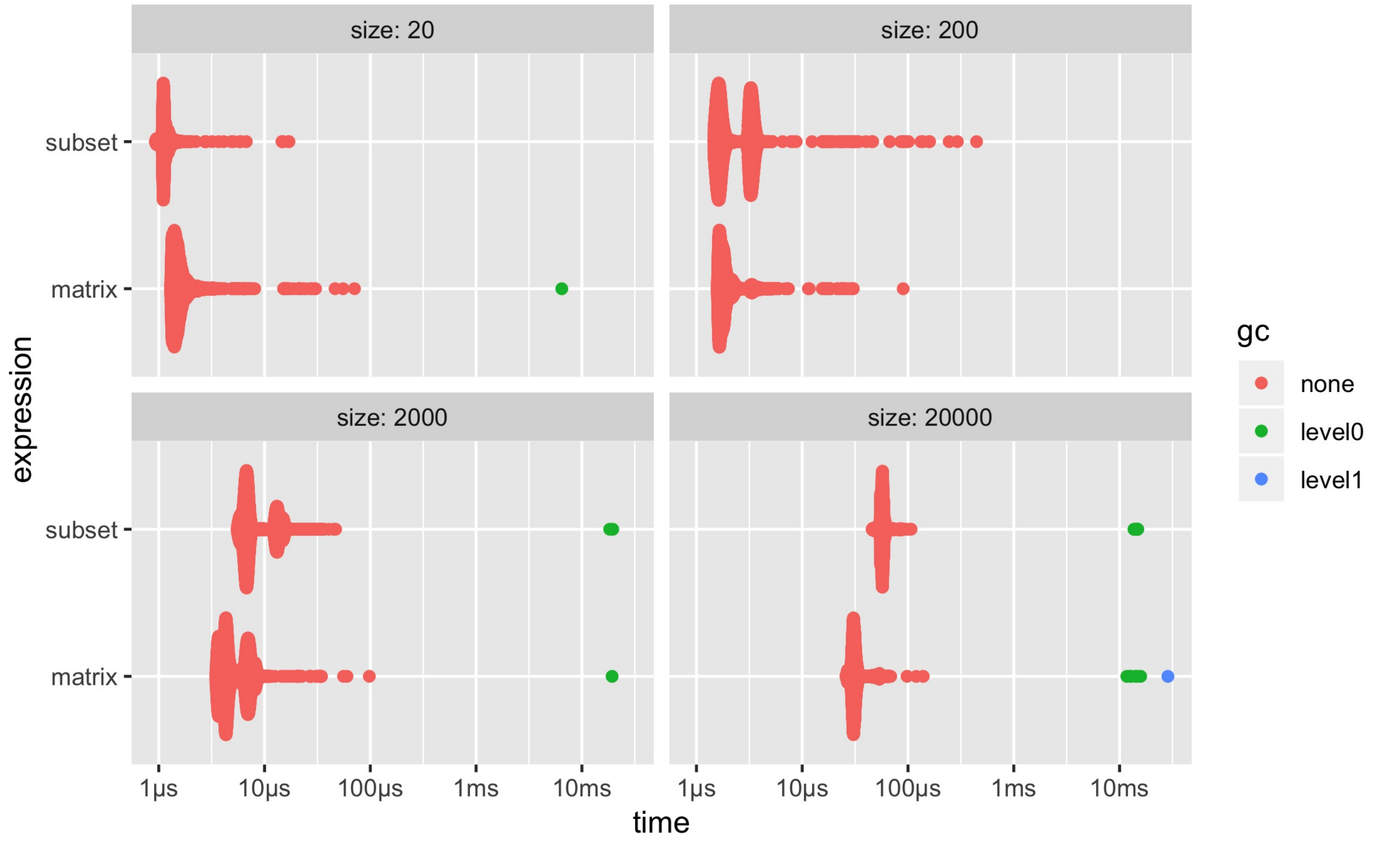
```
x <- matrix(rnorm(20), ncol = 2)
bench::mark(
  subset = x[, 1, drop = FALSE] + x[, 2, drop = FALSE],
  matrix = {
    y <- matrix(c(1, 1), ncol = 1)
    x %*% y
  }
)[c("expression", "min", "mean", "max", "itr/sec")]
```

```
## # A tibble: 2 x 5
##   expression      min     mean      max `itr/sec`
##   <chr>      <bch:tm> <bch:tm> <bch:tm>     <dbl>
## 1 subset       1.02µs   1.37µs    229µs   729534.
## 2 matrix       1.45µs   1.79µs   12.5µs   559247.
```

Use matrix algebra

```
bench::press(
  size = c(20, 200, 2000, 20000),
  {
    x <- matrix(rnorm(size), ncol = 2)
    bench::mark(
      matrix = {
        y <- matrix(c(1, 1), ncol = 1)
        x %*% y
      },
      subset = x[, 1, drop = FALSE] + x[, 2, drop = FALSE]
    )
  }
) %>%
  plot()
```

Use matrix algebra



Size Matters

Size Matters

Sometimes

Size Matters

Sometimes

Always benchmark changes

Size Matters

Sometimes

Always benchmark changes

Save all attempts

Size Matters

Sometimes

Always benchmark changes

Save all attempts