andzinskihw6 benchmark vignette

Maciej Andzinski

June 25, 2015

This document contains results of benchmark of the functions provided by andzinskihw6 package.

Contents

1 Introduction

Package `andzinskihw6` is loaded by

```
library(andzinskihw6)

##
## Attaching package:  'andzinskihw6'
##
## The following objects are masked from 'package:base':
##
##    mode, simplify2array
```

This vignette provides some information about performance of `andzinskihw6` functions.

2 mode() function

For benchmark purposes function `mode2()` was coded in plain R language. It mimics behaviour of `andzinskihw6::mode()` function:

```
mode2 <- function(x) {
  which.max(table(x))
}
```

Benchmark results are presented below:

```
microbenchmark::microbenchmark(
  mode(c(1:100000)),
  mode2(c(1:100000)),
  times=10
)
```

```
## Unit: milliseconds
##                expr       min       lq      mean    median        uq
##   mode(c(1:1e+05))  31.31993  31.9044  35.74104  33.57874  36.70972
##  mode2(c(1:1e+05)) 147.90201 200.5321 213.90075 205.53471 210.83667
##         max neval
##    51.98479    10
##   391.58294    10
```

```
microbenchmark::microbenchmark(
  mode(c(1:10000)),
  mode2(c(1:10000)),
  times=100
)
```

```
## Unit: milliseconds
##                expr      min        lq      mean   median        uq
##   mode(c(1:10000)) 2.273893  2.311374  2.732316  2.46268  2.713628
##  mode2(c(1:10000)) 9.938375 10.229245 11.198887 11.06648 11.752391
##         max neval
```

```
##   6.003443   100
##  14.060294   100
```

```
microbenchmark::microbenchmark(
  mode(c(1:10)),
  mode2(c(1:10)),
  times=100
)

## Unit: microseconds
##            expr     min       lq      mean   median       uq      max
##   mode(c(1:10))  11.094  11.9355  13.41318  13.4145  14.3175   38.055
##  mode2(c(1:10)) 238.485 243.1655 271.59006 248.6625 268.3950 1143.819
##  neval
##    100
##    100
```

As you can see, independently of input size function `andzinskihw6::mode()` is much faster that simmilar solution in R.

3 simplify2array() function

This function provides incomplete functionality comparing to `base::simplify2array`, however it results in slightly faster execution in some cases. Below you can see comparison.

```
microbenchmark::microbenchmark(
  simplify2array(list(c(1:1000),c(1:1000))),
  base::simplify2array(list(c(1:1000),c(1:1000))),
  times=100
)

## Unit: microseconds
##                                              expr   min      lq      mean
##       simplify2array(list(c(1:1000), c(1:1000))) 38.79 40.3335  42.70545
##  base::simplify2array(list(c(1:1000), c(1:1000))) 82.26 84.2310 100.72057
##   median      uq      max neval
##  41.5185 42.8340   86.913   100
##  85.1205 86.6955 1090.380   100
```

Unfortunetly, much bigger input causes significant drop in execution time:

```
microbenchmark::microbenchmark(
  simplify2array(list(c(1:100000),c(1:100000))),
  base::simplify2array(list(c(1:1000),c(1:1000))),
  times=100
)

## Unit: microseconds
##                                                 expr      min       lq
##      simplify2array(list(c(1:1e+05), c(1:1e+05))) 2724.076 4129.297
##  base::simplify2array(list(c(1:1000), c(1:1000)))   84.132   87.798
##       mean   median        uq       max neval
##  6101.3018 4863.451 5436.4715 48617.808   100
##   152.1586  108.153  191.1675  1276.114   100
```

However, for small datasets a little gain is still observed:

```
microbenchmark::microbenchmark(
  simplify2array(list(c(1:10000))),
  base::simplify2array(list(c(1:10000))) ,
  times=100
)

## Unit: microseconds
##                                    expr     min       lq     mean   median
##        simplify2array(list(c(1:10000))) 112.051 148.6305 199.9370 152.6520
##  base::simplify2array(list(c(1:10000))) 143.778 152.9115 207.3481 157.7925
##        uq      max neval
##  155.4270 1762.447   100
##  176.6055 2319.811   100
```

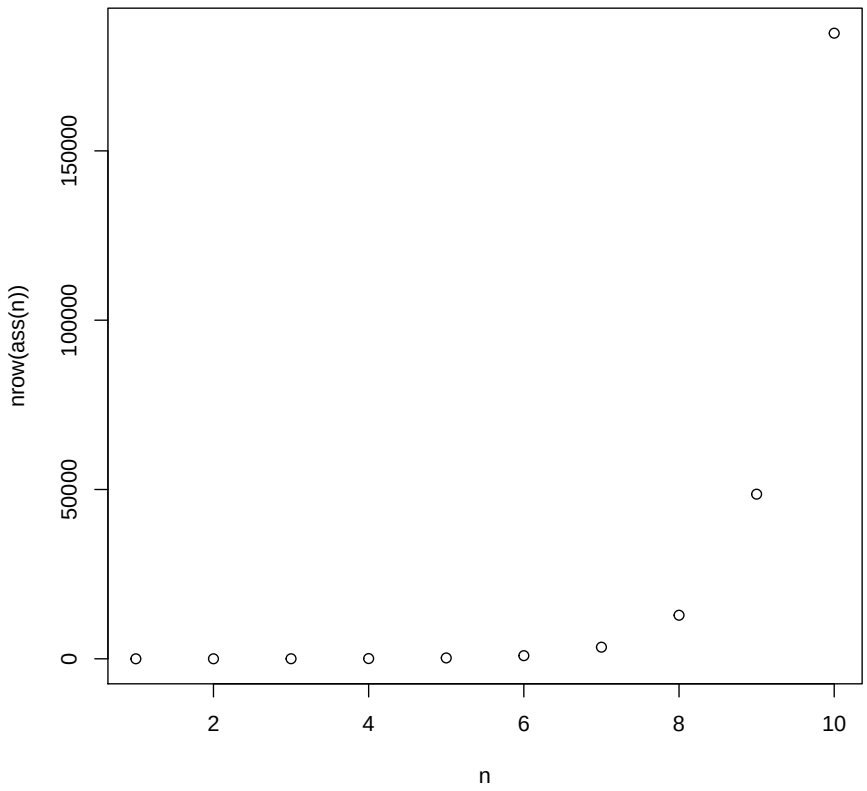
4 ass() function

Below you can see number of rows produced by `ass()` function for `n = 1..10`.

```r
n <- sapply(c(1:10), function(x) { nrow(ass(x)) })
names(n) <- c(1:10)
as.matrix(n)
```

```
##       [,1]
## 1        2
## 2        6
## 3       20
## 4       70
## 5      252
## 6      924
## 7     3432
## 8    12870
## 9    48620
## 10 184756
```

```r
plot(n, xlab="n", ylab="nrow(ass(n))")
```



As you can see, the output grows exponentially. For `n=13` number of rows exceeds 10M (output matrix size is about 2GB), so be prepared for waiting and make sure that your system is equipped with sufficient amount of memory. Due to constraints `sizeof(unsigned long n)`, `n` can't be bigger than 30.

```
n <- sapply(c(1:10), function(x) { object.size(ass(x)) })
names(n) <- c(1:10)
as.matrix(n)
```

```
##        [,1]
## 1       144
## 2       304
## 3      1072
## 4      4592
```

```
## 5     20272
## 6     88816
## 7    384496
## 8   1647472
## 9   7001392
## 10 29561072
```

```
plot(n, xlab="n", ylab="object.size(ass(n)) [bytes]")
```