SortTimes

Samyak Ahuja

August 3, 2018

Complexity for different Sorting Algorithms.

Helper Functions

Replicator

```
replicator <- function(func){
  ele <- seq(from = 0, to = 10000, by = 250)
  ele <- ele[-1]
  timeElapsed <- c()
  for(n in ele){
    timeElapsed <- c(timeElapsed, system.time(replicate(10, func(sample(x = 1:100, size = n, replace =
  }
  return (data.frame(timeElapsed,ele))
}
```

Plotter

```
plotter <- function(df){
  ggplot(df, aes(timeElapsed, ele, color = timeElapsed)) +
    geom_point(shape = 16, size = 5, show.legend = FALSE, alpha = 0.6) +
    theme_minimal() +
    scale_color_gradient(low = "#32aeff", high = "#f2aeff")
}
```

Insertion Sort

Sorting Algorithm

```
insertionSort <- function(vec){
  n <- length(vec)
  for(i in 2:n){
    val <- vec[i]
    pos <- which.max(vec[1:i] > val) #returns index of first occurence of TRUE
    if(pos == 1){
      if(val < vec[1]){
        vec <- c(val, vec[-i])
      }
    }
    else{
      vec <- vec[-i]
      vec <- c(vec[1:(pos-1)], val, vec[pos:(n-1)])
    }
  }
  return (vec)
}
```

Proof of concept

```
insertionSort(c(1,2,99,-21,2,23,1))
```

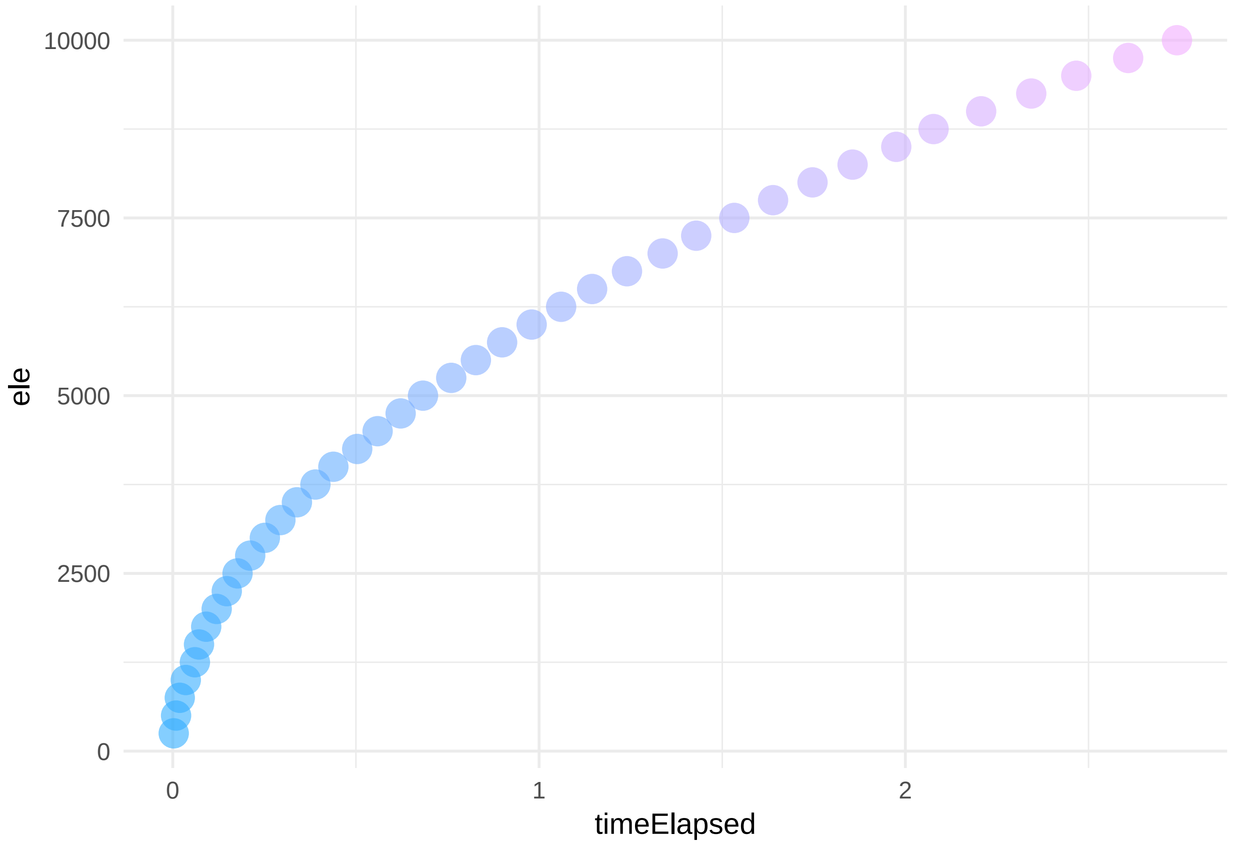
```
## [1] -21   1   1   2   2  23  99
```

RunTime and Plot

```
isdf <- replicator(insertionSort)
isdf
```

```
##    timeElapsed  ele
## 1       0.0026  250
## 2       0.0090  500
## 3       0.0191  750
## 4       0.0355 1000
## 5       0.0604 1250
## 6       0.0716 1500
## 7       0.0913 1750
## 8       0.1199 2000
## 9       0.1474 2250
## 10      0.1768 2500
## 11      0.2114 2750
## 12      0.2514 3000
## 13      0.2938 3250
## 14      0.3389 3500
## 15      0.3894 3750
## 16      0.4384 4000
## 17      0.5035 4250
## 18      0.5592 4500
## 19      0.6221 4750
## 20      0.6831 5000
## 21      0.7602 5250
## 22      0.8275 5500
## 23      0.8991 5750
## 24      0.9798 6000
## 25      1.0602 6250
## 26      1.1448 6500
## 27      1.2400 6750
## 28      1.3373 7000
## 29      1.4290 7250
## 30      1.5330 7500
## 31      1.6388 7750
## 32      1.7465 8000
## 33      1.8559 8250
## 34      1.9751 8500
## 35      2.0769 8750
## 36      2.2069  9000
## 37      2.3436  9250
## 38      2.4665  9500
## 39      2.6082  9750
## 40      2.7414 10000
```

```
plotter(isdf)
```



Merge Sort

Sorting Algorithm

```
mergeSort <- function(vec){

  mergeTwo <- function(left,right){
    res <- c()
    while(length(left) > 0 && length(right) > 0){
      if(left[1] <= right[1]){
        res <- c(res,left[1])
        left <- left[-1]
      }else{
        res <- c(res,right[1])
        right <- right[-1]
      }
    }
    if(length(left) > 0) res <- c(res,left)
    if(length(right) > 0) res <- c(res,right)
    return (res)
  }

  n <- length(vec)
  if(n <= 1) return (vec)
  else{
    middle <- length(vec) / 2
    left <- vec[1:floor(middle)]
    right <- vec[floor(middle + 1):n]
    left <- mergeSort(left)
    right <- mergeSort(right)
    if(left[length(left)] <= right[1]){
      return (c(left,right))
    }else{
      return (mergeTwo(left,right))
    }
  }
}
```

Proof of Concept

```
mergeSort(c(12,-22,13,2,-33,2))
```

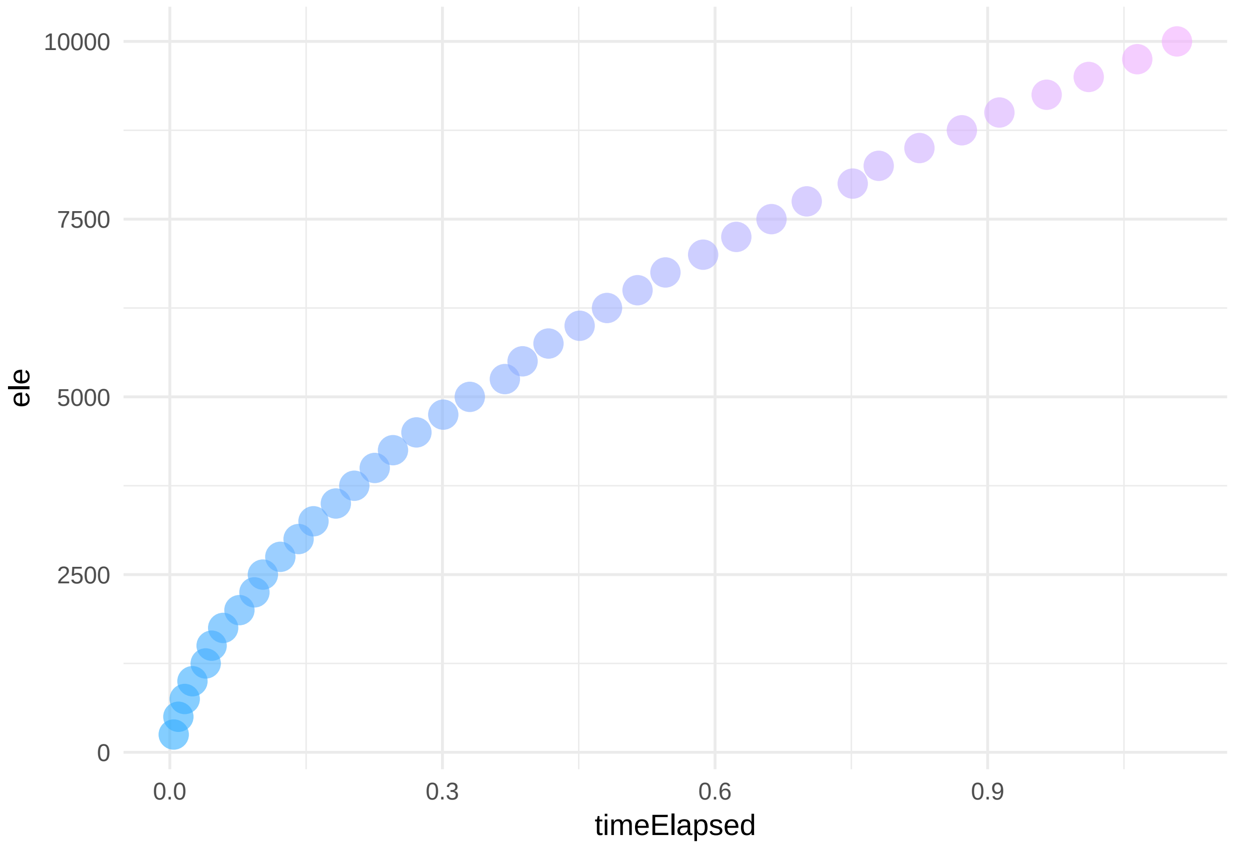
```
## [1] -33 -22   2   2  12  13
```

RunTime and Plot

```
msdf <- replicator(mergeSort)
msdf
```

```
##    timeElapsed   ele
## 1       0.0043   250
## 2       0.0094   500
## 3       0.0164   750
## 4       0.0249  1000
## 5       0.0395  1250
## 6       0.0460  1500
## 7       0.0587  1750
## 8       0.0766  2000
## 9       0.0931  2250
## 10      0.1024  2500
## 11      0.1216  2750
## 12      0.1417  3000
## 13      0.1581  3250
## 14      0.1827  3500
## 15      0.2030  3750
## 16      0.2255  4000
## 17      0.2456  4250
## 18      0.2714  4500
## 19      0.3009  4750
## 20      0.3301  5000
## 21       0.3687  5250
## 22       0.3882  5500
## 23       0.4167  5750
## 24       0.4510  6000
## 25       0.4811  6250
## 26       0.5147  6500
## 27       0.5454  6750
## 28       0.5869  7000
## 29       0.6234  7250
## 30       0.6621  7500
## 31       0.7009  7750
## 32       0.7515  8000
## 33       0.7801  8250
## 34       0.8249  8500
## 35       0.8716  8750
## 36       0.9129  9000
## 37       0.9649  9250
## 38       1.0112  9500
## 39       1.0646  9750
## 40       1.1083 10000
```

```
plotter(msdf)
```



Quick Sort

Sorting Algorithm

```
quickSort <- function(vec){
  if(length(vec) > 1){
    pivot <- median(vec)
    return (c(quickSort(vec[vec < pivot]), vec[vec == pivot], quickSort(vec[vec > pivot])))
  }else{
    return (vec)
  }
}
```

Proof of Concept

```
quickSort(c(12,-22,13,2,-33,2))
```

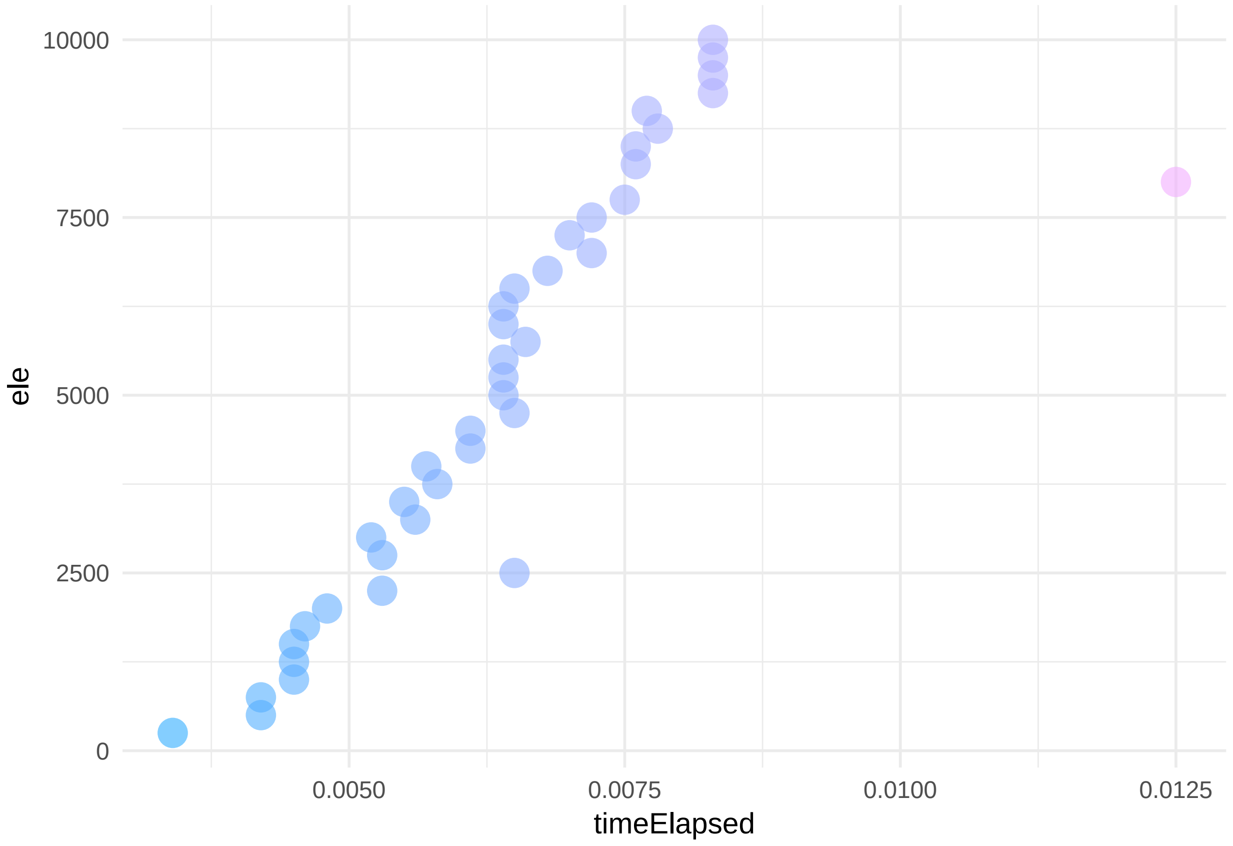
```
## [1] -33 -22   2   2  12  13
```

RunTime and Plot

```
qsdf <- replicator(quickSort)
qsdf
```

```
##    timeElapsed   ele
## 1       0.0034   250
## 2       0.0042   500
## 3       0.0042   750
## 4       0.0045  1000
## 5       0.0045  1250
## 6       0.0045  1500
## 7       0.0046  1750
## 8       0.0048  2000
## 9       0.0053  2250
## 10      0.0065  2500
## 11      0.0053  2750
## 12      0.0052  3000
## 13      0.0056  3250
## 14      0.0055  3500
## 15      0.0058  3750
## 16      0.0057  4000
## 17      0.0061  4250
## 18      0.0061  4500
## 19      0.0065  4750
## 20      0.0064  5000
## 21      0.0064  5250
## 22      0.0064  5500
## 23      0.0066  5750
## 24      0.0064  6000
## 25      0.0064  6250
## 26      0.0065  6500
## 27       0.0068  6750
## 28       0.0072  7000
## 29       0.0070  7250
## 30       0.0072  7500
## 31       0.0075  7750
## 32       0.0125  8000
## 33       0.0076  8250
## 34       0.0076  8500
## 35       0.0078  8750
## 36       0.0077  9000
## 37       0.0083  9250
## 38       0.0083  9500
## 39       0.0083  9750
## 40       0.0083 10000
```

```
plotter(qsdf)
```


ele
timeElapsed