# SortTimes

*Samyak Ahuja*

*August 3, 2018*

## Complexity for different Sorting Algorithms.

### Helper Functions

#### Replicator

```r
replicator <- function(func, size = 1000){
  if(size == 1000){
      ele <- seq(from = 0, to = 1000, by = 50)
  }else{
      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 = TRUE)))
                       )[3] / 10
                     )
  }
  return (data.frame(ele,timeElapsed))
}
```

#### Plotter

```r
plotter <- function(df, df_title){
  ggplot(df, aes(ele, timeElapsed, color = timeElapsed)) +
    geom_point(shape = 16, size = 5, show.legend = FALSE, alpha = 0.6) +
    stat_smooth(method="loess", formula=y~x) +
    theme_minimal() +
    labs(subtitle = "Time vs Size",
       y = "Time (in seconds)",
       x = "Number of Elements",
       title = df_title) +
    scale_color_gradient(low = "#32aeff", high = "#f2aeff")
}
```

#### Combined Plotter

```r
comb_plotter <- function(df, df_title){
  ggplot(df, aes(ele, value, col = variable)) +
  geom_point(shape = 16, size = 2, alpha = 0.6) +
  stat_smooth(method="loess", formula=y~x) +
  labs(subtitle = "Time vs Size",
       y = "Time (in seconds)",
       x = "Number of Elements",
       title = df_title) +
  stat_poly_eq(parse=T, aes(label = ..eq.label..), formula=y~x)
}
```

## 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))
```

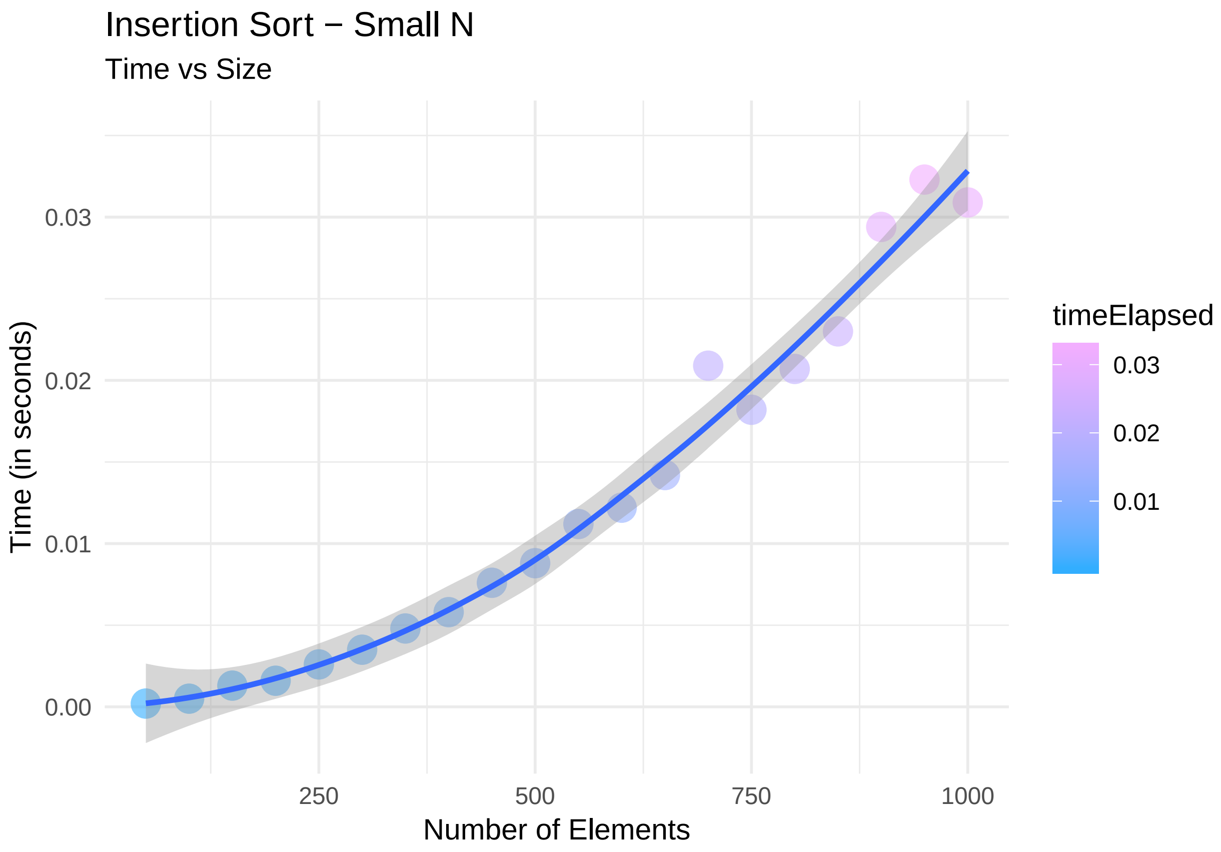
```
## [1] -21   1   1   2   2  23  99
```

### RunTime and Plot

```
isdf_small <- replicator(insertionSort)
isdf_small
```

```
##     ele timeElapsed
## 1    50      0.0002
## 2   100      0.0005
## 3   150      0.0013
## 4   200      0.0016
## 5   250      0.0026
## 6   300      0.0035
## 7   350      0.0048
## 8   400      0.0058
## 9   450       0.0076
## 10  500       0.0088
## 11  550       0.0112
## 12  600       0.0122
## 13  650       0.0142
## 14  700       0.0209
## 15  750       0.0182
## 16  800       0.0207
## 17  850       0.0230
## 18  900       0.0294
## 19  950       0.0323
## 20 1000       0.0309
```

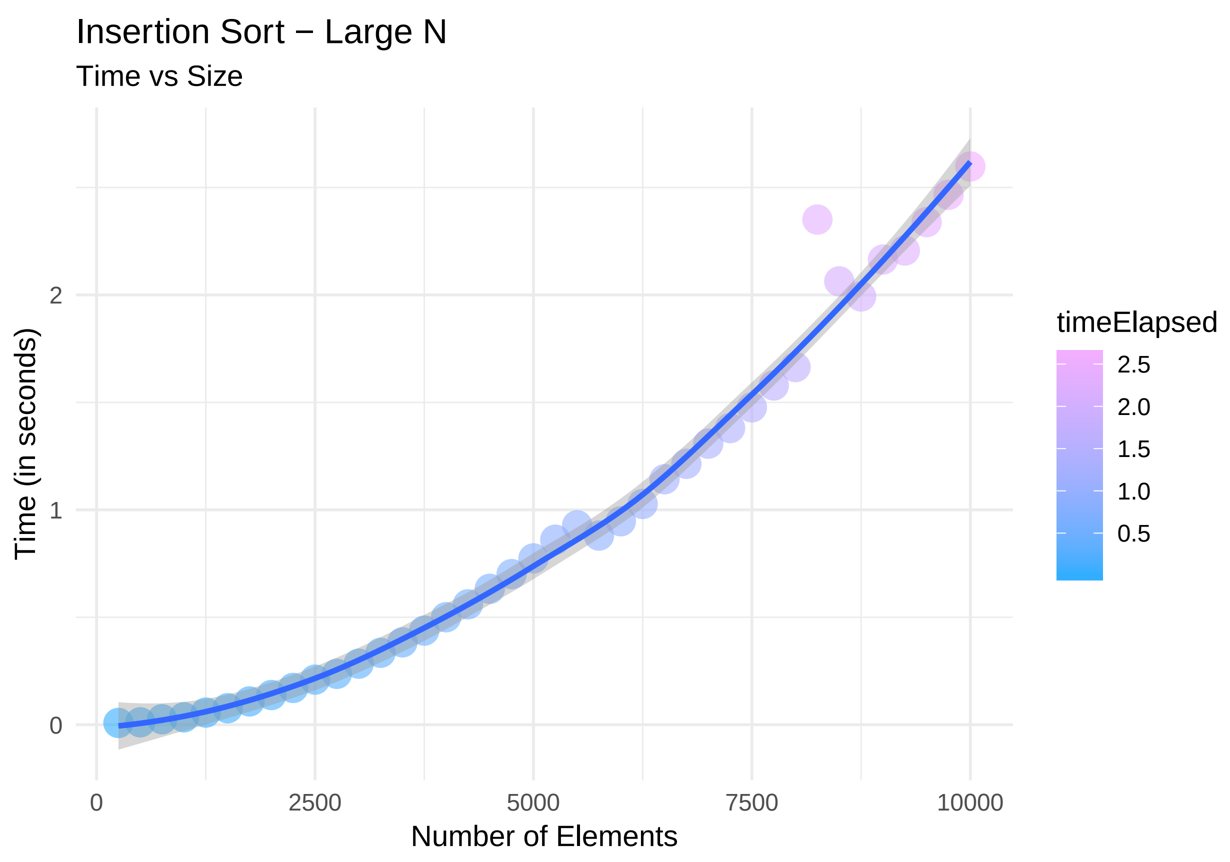
```
plotter(isdf_small, "Insertion Sort - Small N")
```



```
isdf_big <- replicator(insertionSort, 10000)
isdf_big
```

```
##      ele timeElapsed
## 1    250      0.0082
## 2    500      0.0112
## 3    750      0.0254
## 4   1000      0.0344
## 5   1250      0.0565
## 6   1500      0.0767
## 7   1750      0.1083
## 8   2000      0.1380
## 9   2250      0.1711
## 10  2500       0.2097
## 11  2750       0.2368
## 12  3000       0.2841
## 13  3250       0.3348
## 14  3500       0.3834
## 15  3750       0.4380
## 16  4000       0.5002
## 17  4250       0.5603
## 18  4500       0.6325
## 19  4750       0.7008
## 20  5000       0.7741
## 21  5250       0.8609
## 22  5500       0.9281
## 23  5750       0.8802
## 24  6000       0.9468
## 25  6250       1.0273
## 26  6500       1.1423
## 27  6750       1.2141
## 28  7000       1.3080
## 29  7250       1.3804
## 30  7500       1.4768
## 31  7750       1.5786
## 32  8000       1.6643
## 33  8250       2.3508
## 34  8500       2.0633
## 35  8750       1.9929
## 36  9000       2.1649
## 37  9250       2.2068
## 38  9500       2.3389
## 39  9750       2.4663
## 40 10000       2.5979
```

```
plotter(isdf_big, "Insertion Sort - Large N")
```



## 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))
```

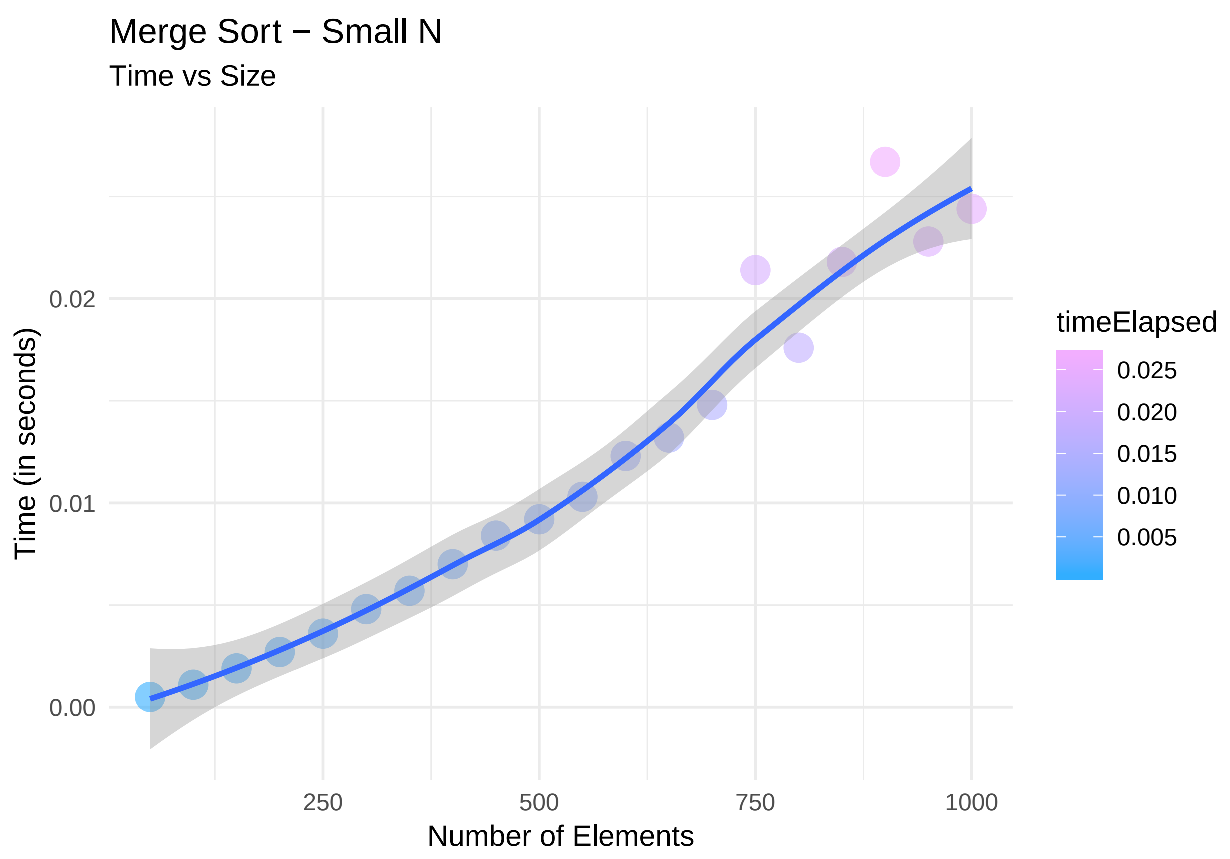
```
## [1] -33 -22   2   2  12  13
```

### RunTime and Plot

```
msdf_small <- replicator(mergeSort)
msdf_small
```

```
##     ele timeElapsed
## 1    50      0.0005
## 2   100      0.0011
## 3   150      0.0019
## 4   200      0.0027
## 5   250      0.0036
## 6   300      0.0048
## 7   350      0.0057
## 8   400      0.0070
## 9   450      0.0084
## 10  500      0.0092
## 11  550      0.0103
## 12  600      0.0123
## 13  650      0.0132
## 14  700      0.0148
## 15  750      0.0214
## 16  800      0.0176
## 17  850      0.0218
## 18  900      0.0267
## 19  950      0.0228
## 20 1000      0.0244
```

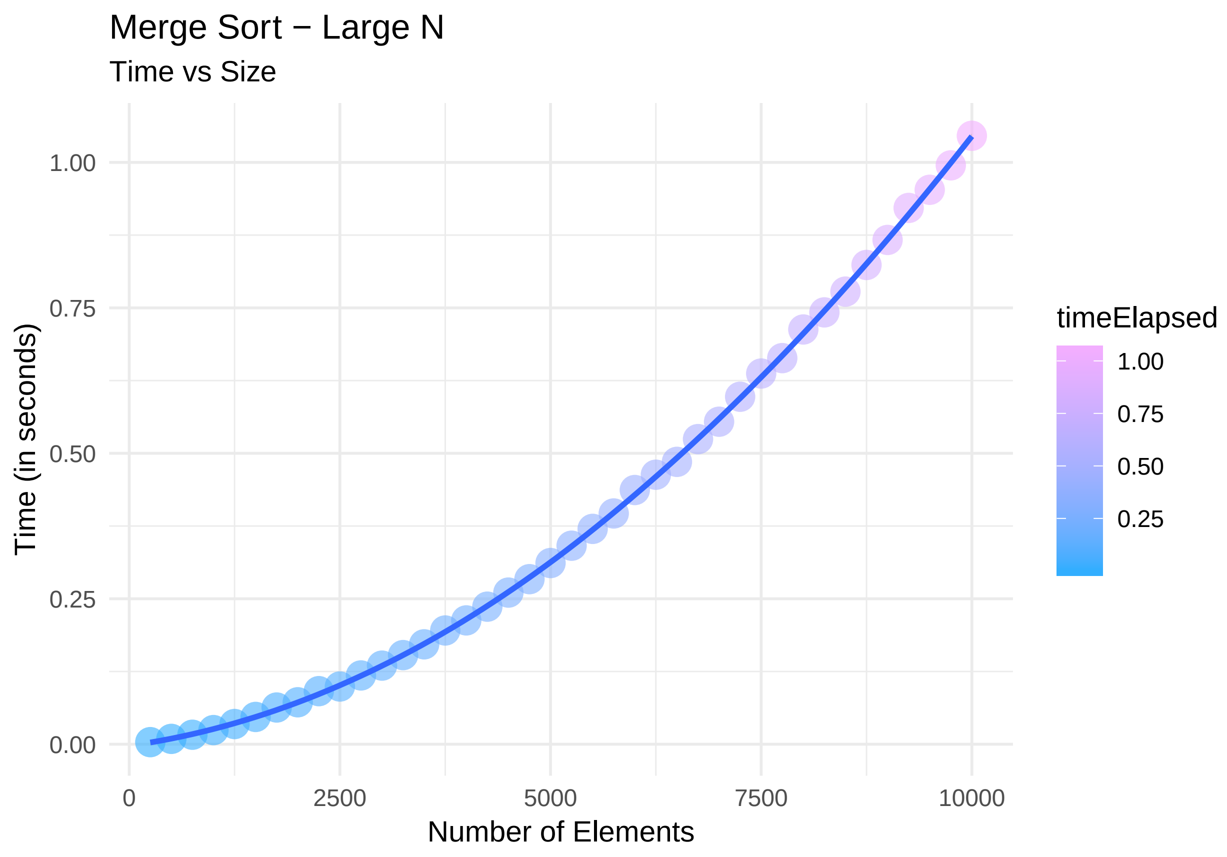
```
plotter(msdf_small, "Merge Sort - Small N")
```



```r
msdf_big <- replicator(mergeSort, 10000)
msdf_big
```

```
##       ele timeElapsed
## 1     250      0.0036
## 2     500      0.0092
## 3     750      0.0163
## 4    1000      0.0243
## 5    1250      0.0347
## 6    1500      0.0469
## 7    1750      0.0631
## 8    2000      0.0721
## 9    2250      0.0912
## 10   2500      0.0992
## 11   2750      0.1182
## 12   3000      0.1351
## 13   3250      0.1532
## 14   3500      0.1717
## 15   3750      0.1955
## 16   4000      0.2128
## 17   4250      0.2364
## 18   4500      0.2606
## 19   4750      0.2837
## 20   5000      0.3114
## 21   5250      0.3412
## 22   5500      0.3701
## 23   5750      0.3966
## 24  6000       0.4369
## 25  6250       0.4632
## 26  6500       0.4853
## 27  6750       0.5244
## 28  7000       0.5543
## 29  7250       0.5972
## 30  7500       0.6371
## 31  7750       0.6636
## 32  8000       0.7128
## 33  8250       0.7422
## 34  8500       0.7779
## 35  8750       0.8237
## 36  9000       0.8667
## 37  9250       0.9217
## 38  9500       0.9529
## 39  9750       0.9949
## 40 10000       1.0457
```

```
plotter(msdf_big, "Merge Sort - Large N")
```



## 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))
```

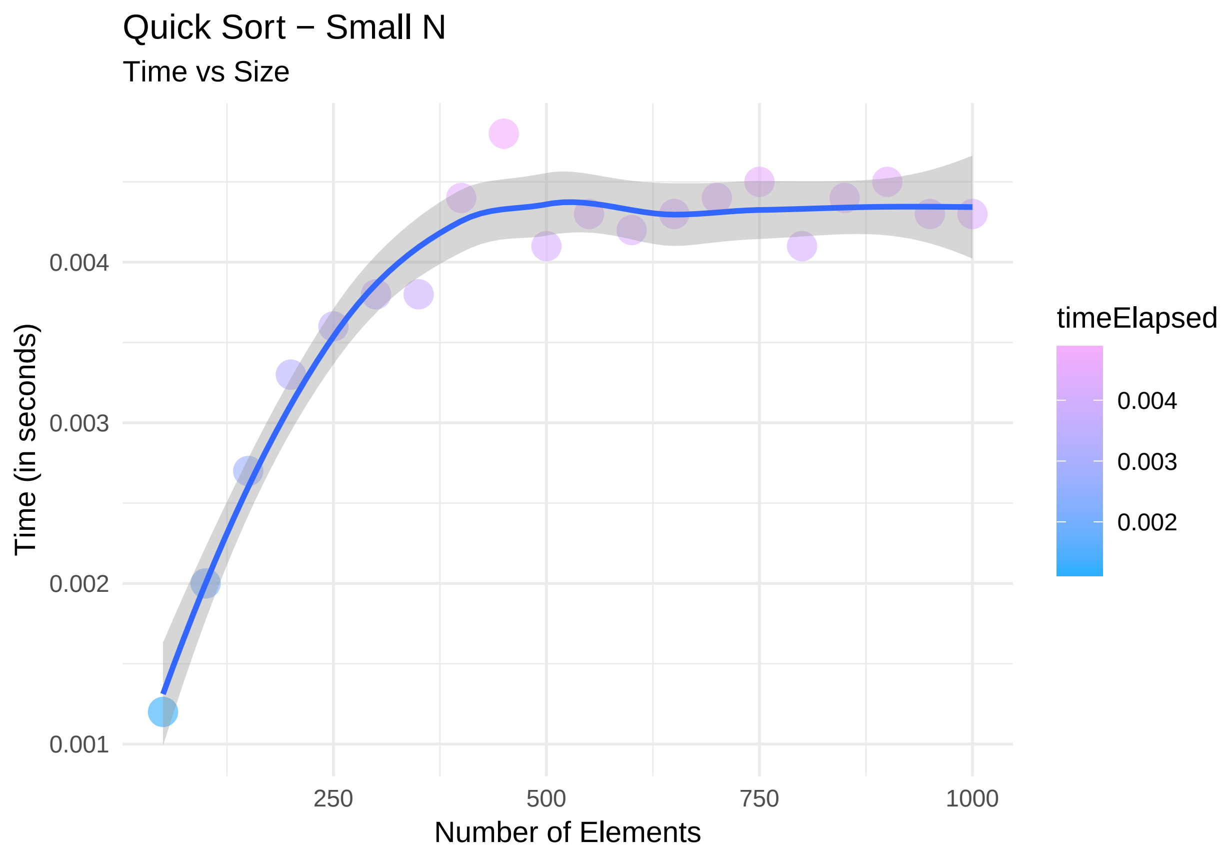
```
## [1] -33 -22   2   2  12  13
```

### RunTime and Plot

```
qsdf_small <- replicator(quickSort)
qsdf_small
```

```
##     ele timeElapsed
## 1    50      0.0012
## 2   100      0.0020
## 3   150      0.0027
## 4   200      0.0033
## 5   250      0.0036
## 6   300      0.0038
## 7   350      0.0038
## 8   400      0.0044
## 9   450      0.0048
## 10  500      0.0041
## 11  550      0.0043
## 12  600      0.0042
## 13  650      0.0043
## 14  700      0.0044
## 15  750      0.0045
## 16  800      0.0041
## 17  850      0.0044
## 18  900      0.0045
## 19  950      0.0043
## 20 1000      0.0043
```

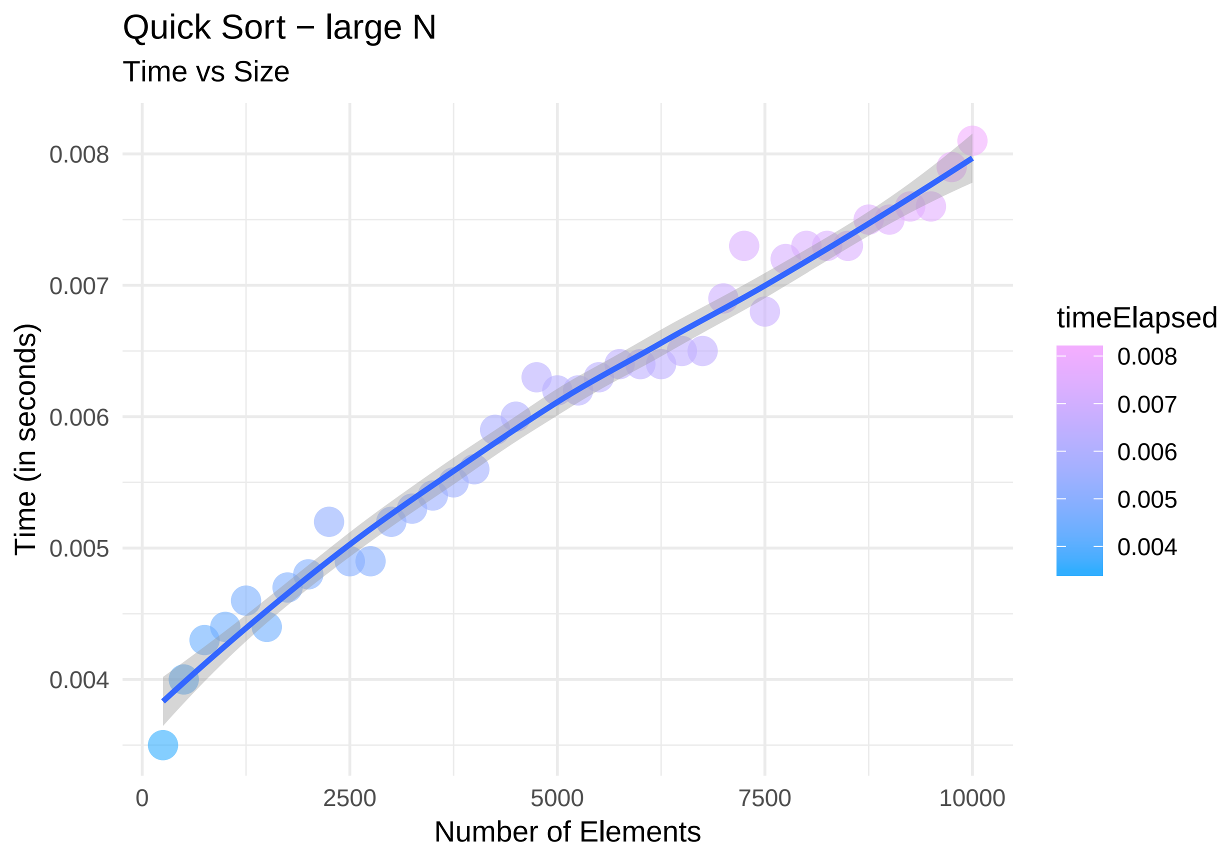
```
plotter(qsdf_small, "Quick Sort - Small N")
```



```r
qsdf_big <- replicator(quickSort, 10000)
qsdf_big
```

```
##       ele timeElapsed
## 1     250      0.0035
## 2     500      0.0040
## 3     750      0.0043
## 4    1000      0.0044
## 5    1250      0.0046
## 6    1500      0.0044
## 7    1750      0.0047
## 8    2000      0.0048
## 9    2250      0.0052
## 10   2500      0.0049
## 11   2750      0.0049
## 12   3000      0.0052
## 13   3250      0.0053
## 14   3500      0.0054
## 15   3750      0.0055
## 16   4000      0.0056
## 17   4250      0.0059
## 18   4500      0.0060
## 19   4750      0.0063
## 20   5000      0.0062
## 21   5250      0.0062
## 22   5500      0.0063
## 23   5750      0.0064
## 24  6000       0.0064
## 25  6250       0.0064
## 26  6500       0.0065
## 27  6750       0.0065
## 28  7000       0.0069
## 29  7250       0.0073
## 30  7500       0.0068
## 31  7750       0.0072
## 32  8000       0.0073
## 33  8250       0.0073
## 34  8500       0.0073
## 35  8750       0.0075
## 36  9000       0.0075
## 37  9250       0.0076
## 38  9500       0.0076
## 39  9750       0.0079
## 40 10000       0.0081
```

```
plotter(qsdf_big, "Quick Sort - large N")
```



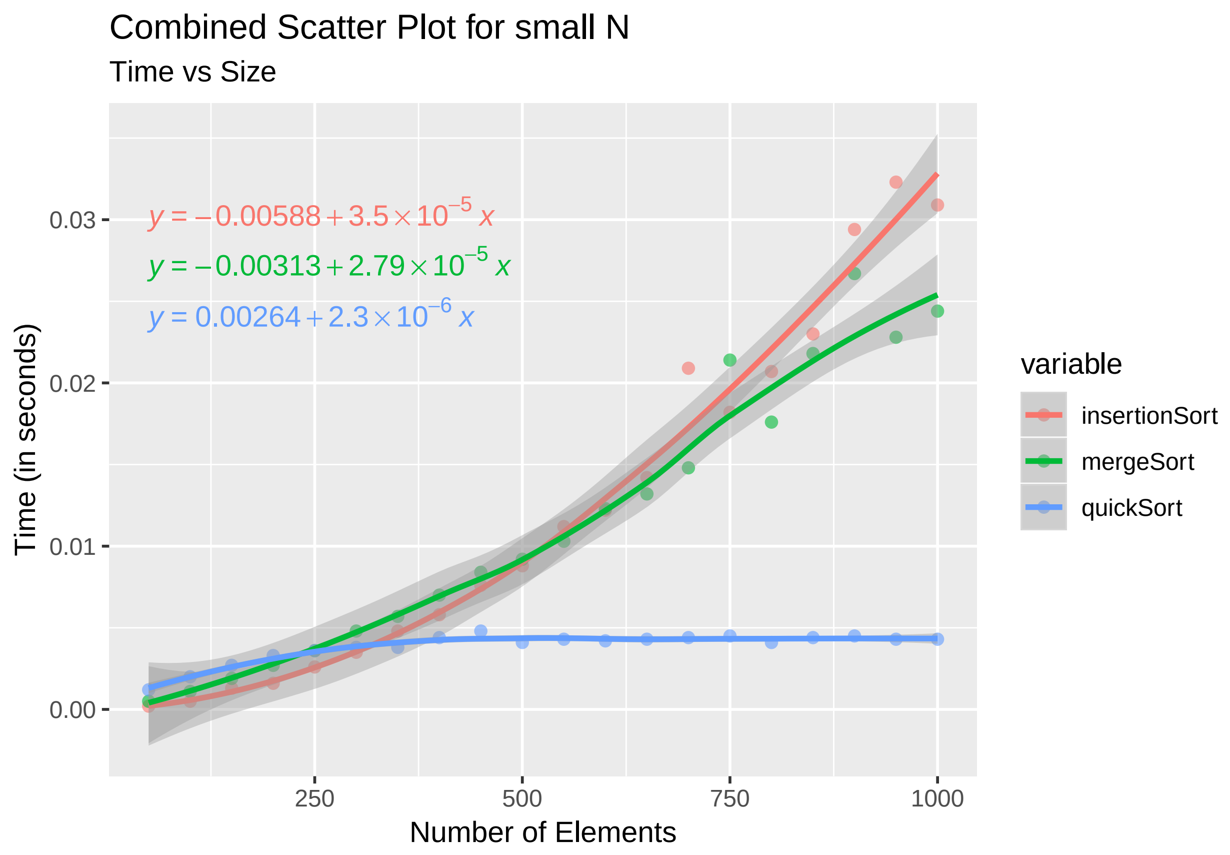
## Combined Plots

### Small N

```
df_small <- data.frame(insertionSort = isdf_small[[2]],
                       mergeSort = msdf_small[[2]],
                       quickSort = qsdf_small[[2]],
                       ele = msdf_small[[1]])
df_small
```

```
##    insertionSort mergeSort quickSort  ele
## 1         0.0002    0.0005    0.0012   50
## 2         0.0005    0.0011    0.0020  100
## 3         0.0013    0.0019    0.0027  150
## 4         0.0016    0.0027    0.0033  200
## 5         0.0026    0.0036    0.0036  250
## 6         0.0035    0.0048    0.0038  300
## 7         0.0048    0.0057    0.0038  350
## 8         0.0058    0.0070    0.0044  400
## 9         0.0076    0.0084    0.0048  450
## 10        0.0088    0.0092    0.0041  500
## 11        0.0112    0.0103    0.0043  550
## 12        0.0122    0.0123    0.0042  600
## 13        0.0142    0.0132    0.0043  650
## 14        0.0209    0.0148    0.0044  700
## 15        0.0182    0.0214    0.0045  750
## 16        0.0207    0.0176    0.0041  800
## 17        0.0230    0.0218    0.0044  850
## 18        0.0294    0.0267    0.0045  900
## 19        0.0323    0.0228    0.0043  950
## 20        0.0309    0.0244    0.0043 1000
```

```
df_small <- melt(df_small, id.vars = "ele")
comb_plotter(df_small, "Combined Scatter Plot for small N")
```

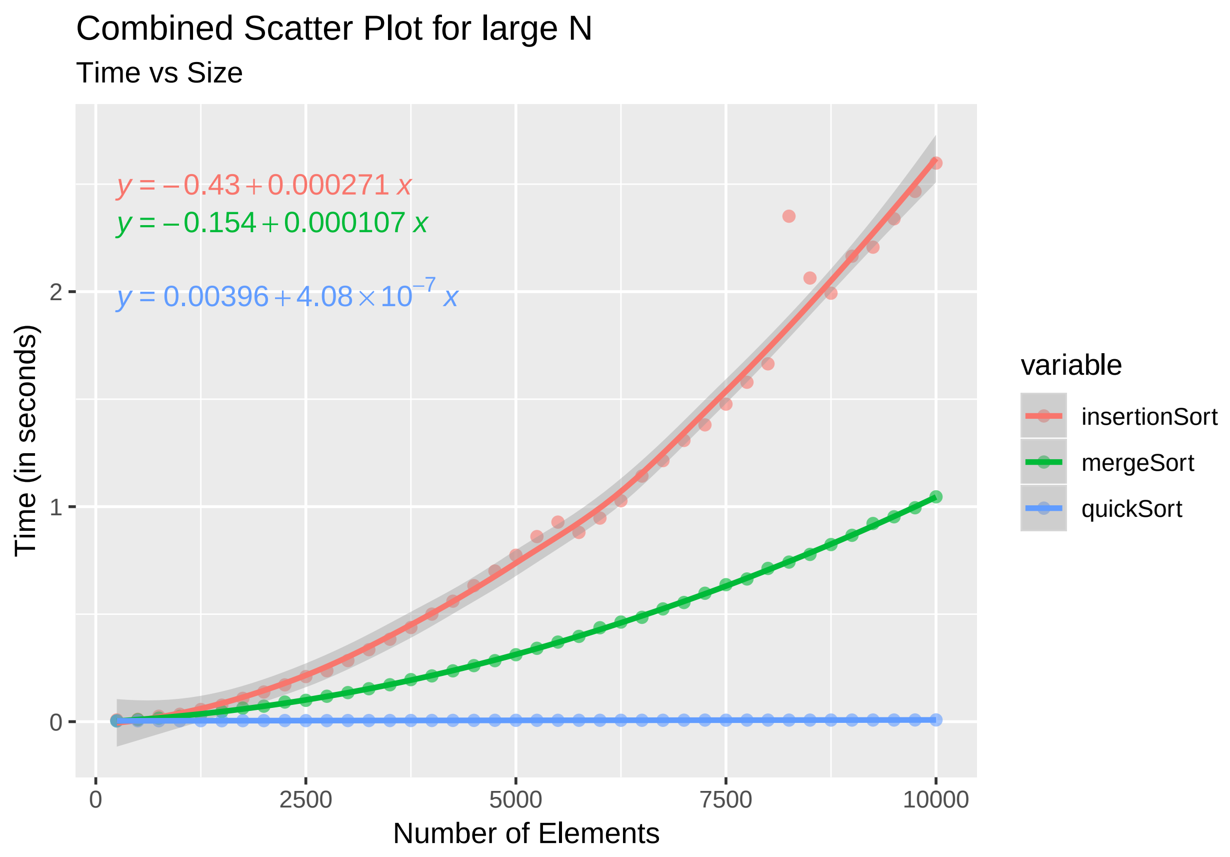


**Large N**

```
df_big <- data.frame(insertionSort = isdf_big[[2]],
                 mergeSort = msdf_big[[2]],
                 quickSort = qsdf_big[[2]],
                 ele = msdf_big[[1]])
df_big
```

```
##    insertionSort mergeSort quickSort  ele
## 1         0.0082    0.0036    0.0035  250
## 2         0.0112    0.0092    0.0040  500
## 3         0.0254    0.0163    0.0043  750
## 4         0.0344    0.0243    0.0044 1000
## 5         0.0565    0.0347    0.0046 1250
## 6         0.0767    0.0469    0.0044 1500
## 7         0.1083    0.0631    0.0047 1750
## 8         0.1380    0.0721    0.0048 2000
## 9         0.1711    0.0912    0.0052 2250
## 10        0.2097    0.0992    0.0049 2500
## 11        0.2368    0.1182    0.0049 2750
## 12        0.2841    0.1351    0.0052 3000
## 13        0.3348    0.1532    0.0053 3250
## 14        0.3834    0.1717    0.0054 3500
## 15        0.4380    0.1955    0.0055 3750
## 16        0.5002    0.2128    0.0056 4000
## 17        0.5603    0.2364    0.0059  4250
## 18        0.6325    0.2606    0.0060  4500
## 19        0.7008    0.2837    0.0063  4750
## 20        0.7741    0.3114    0.0062  5000
## 21        0.8609    0.3412    0.0062  5250
## 22        0.9281    0.3701    0.0063  5500
## 23        0.8802    0.3966    0.0064  5750
## 24        0.9468    0.4369    0.0064  6000
## 25        1.0273    0.4632    0.0064  6250
## 26        1.1423    0.4853    0.0065  6500
## 27        1.2141    0.5244    0.0065  6750
## 28        1.3080    0.5543    0.0069  7000
## 29        1.3804    0.5972    0.0073  7250
## 30        1.4768    0.6371    0.0068  7500
## 31        1.5786    0.6636    0.0072  7750
## 32        1.6643    0.7128    0.0073  8000
## 33        2.3508    0.7422    0.0073  8250
## 34        2.0633    0.7779    0.0073  8500
## 35        1.9929    0.8237    0.0075  8750
## 36        2.1649    0.8667    0.0075  9000
## 37        2.2068    0.9217    0.0076  9250
## 38        2.3389    0.9529    0.0076  9500
## 39        2.4663    0.9949    0.0079  9750
## 40        2.5979    1.0457    0.0081 10000
```

```r
df_big <- melt(df_big, id.vars = "ele")
comb_plotter(df_big, "Combined Scatter Plot for large N")
```


Combined Scatter Plot for large N
Time vs Size
$y = -0.43 + 0.000271\,x$
$y = -0.154 + 0.000107\,x$
$y = 0.00396 + 4.08 \times 10^{-7}\,x$
Time (in seconds)
Number of Elements
variable
insertionSort
mergeSort
quickSort