Comparison Analysis for Sorting Algorithms

Samyak Ahuja

Overview

Sorting Algorithms chosen for analysis are :

- Insertion Sort
- Merge Sort
- Quick Sort

Helper Functions

Helper functions are used for two purposes:

a	b	c
1	2	3

Data Generator and Replicator

Data Generator

Objective : To

```
dataSetGenerator <- function(size = 1000, sep = 20){
    ele <- seq(from = 0, to = size, by = sep)
    ele <- ele[-1]
    data <- list()
    for(n in ele){
      iterator <- n / sep
      repeated <- list()
      for(i in 1:10){
        repeated <- c(repeated, list(sample(x = 1:100, size = n, replace = TRUE)))
      }
      data <- c(data, repeated)
    }
    return (data)
}

dataSet <- dataSetGenerator()

replicator <- function(func, size = 1000, sep = 20){
  ele <- seq(from = 0, to = size, by = sep)
  ele <- ele[-1]
  timeElapsed <- c()
  for(n in ele){
    op <- 0
    iterator <- n / sep
```

```
    for(i in 1:10){
         op = op + func(dataSet[[iterator + i]])$operations
    }
    op = op / 10
    timeElapsed <- c(timeElapsed, op)
  }
  return (data.frame(ele,timeElapsed))
}
```

Plotter

plotter function creates a Comparisons vs Elements plot for each sorting algorithm separately.

```
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="lm", formula=y~poly(x,2), rm = FALSE) +
    theme_minimal() +
    labs(subtitle = "Time vs Size",
       y = "Number of Comparisons (Averaged)",
       x = "Number of Elements",
       title = df_title) +
    scale_color_gradient(low = "#32aeff", high = "#f2aeff") +
    stat_poly_eq(parse=T, aes(label = ..eq.label..), formula=y~poly(x,2))
}
```

Combined Plotter

combined plotter function creates a combined Comparisons vs Elements plot for all the sorting algorithms.

```
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="lm", formula=y~poly(x,2)) +
  theme_minimal() +
  labs(subtitle = "Time vs Size",
       y = "Number of Comparisons (Averaged)",
       x = "Number of Elements",
       title = df_title) +
  stat_poly_eq(parse=T, aes(label = ..eq.label..), formula=y~poly(x,2))
}
```

Insertion Sort

Sorting Algorithm

```
insertionSort <- function(vec){
  n <- length(vec)
  op <- 0
  for(i in 2:n){
    key <- vec[i]
```

```
    pos <- i - 1
    while(pos > 0 && vec[pos] > key){
      vec[pos + 1] = vec[pos]
      pos = pos - 1
      op <- op + 1
    }
    vec[pos + 1] <- key
    op <- op + 1
  }
  return (list("vec" = vec, "operations" = op))
}
```

Proof of concept

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

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 14
```

RunTime and Plot

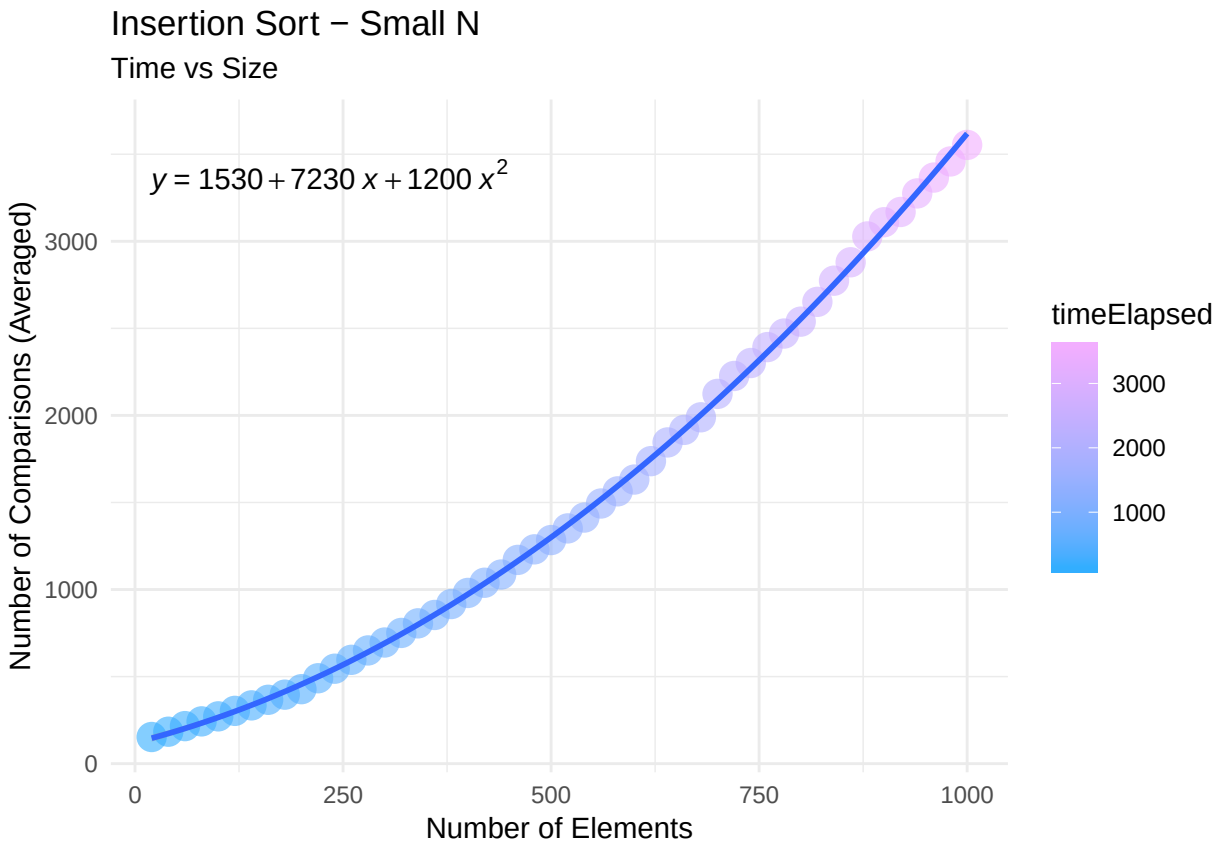
```
isdf_small <- replicator(insertionSort)
isdf_small
```

```
##     ele timeElapsed
## 1    20       153.1
## 2    40       183.5
## 3    60       215.6
## 4    80       242.9
## 5   100       271.7
## 6   120       303.2
## 7   140       334.8
## 8   160       366.8
## 9   180       396.3
## 10  200       427.8
## 11  220       490.5
## 12  240       545.4
## 13  260       596.4
## 14  280       650.2
## 15  300       695.8
## 16  320       750.8
## 17  340       805.9
## 18  360       853.9
## 19  380       916.7
## 20  400       980.4
## 21  420      1039.0
## 22  440      1086.2
## 23  460      1170.2
## 24  480      1230.9
```

```
## 25  500      1284.2
## 26  520      1351.1
## 27  540      1413.4
## 28  560      1494.1
## 29  580      1563.9
## 30  600      1631.7
## 31  620      1737.4
## 32  640      1844.9
## 33  660      1917.0
## 34  680      1989.2
## 35  700      2123.6
## 36  720      2226.5
## 37  740      2300.9
## 38  760      2392.4
## 39  780      2470.2
## 40  800      2538.5
## 41  820      2652.2
## 42  840      2773.7
## 43  860      2879.5
## 44  880      3027.8
## 45  900      3109.4
## 46  920      3168.4
## 47  940      3275.0
## 48  960      3366.0
## 49  980      3458.7
## 50 1000      3553.5
```

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

```
## Warning: Ignoring unknown parameters: rm
```



Merge Sort

Sorting Algorithm

```
mergeSort <- function(vec){

  mergeTwo <- function(left,right){
    op <- 0
    res <- c()
    while(length(left) > 0 && length(right) > 0){
      op <- op + 1
      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)
    }
```

```
    op <- op + 1
    return (list("vec" = res, "operations" = op))
  }

  op <- 0
  n <- length(vec)
  if(n <= 1) return (list("vec" = vec, "operations" = op))
  else{
    middle <- length(vec) %/% 2 #integer division
    left_list <- mergeSort(vec[1:middle])
    right_list <- mergeSort(vec[(middle + 1):n])
    left <- left_list$vec
    right <- right_list$vec
    res <- mergeTwo(left,right)
    op <- op + left_list$operations + right_list$operations + res$operations
    return (list("vec" = res$vec, "operations" = op))
  }
}
```

Proof of Concept

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

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 15
```

RunTime and Plot

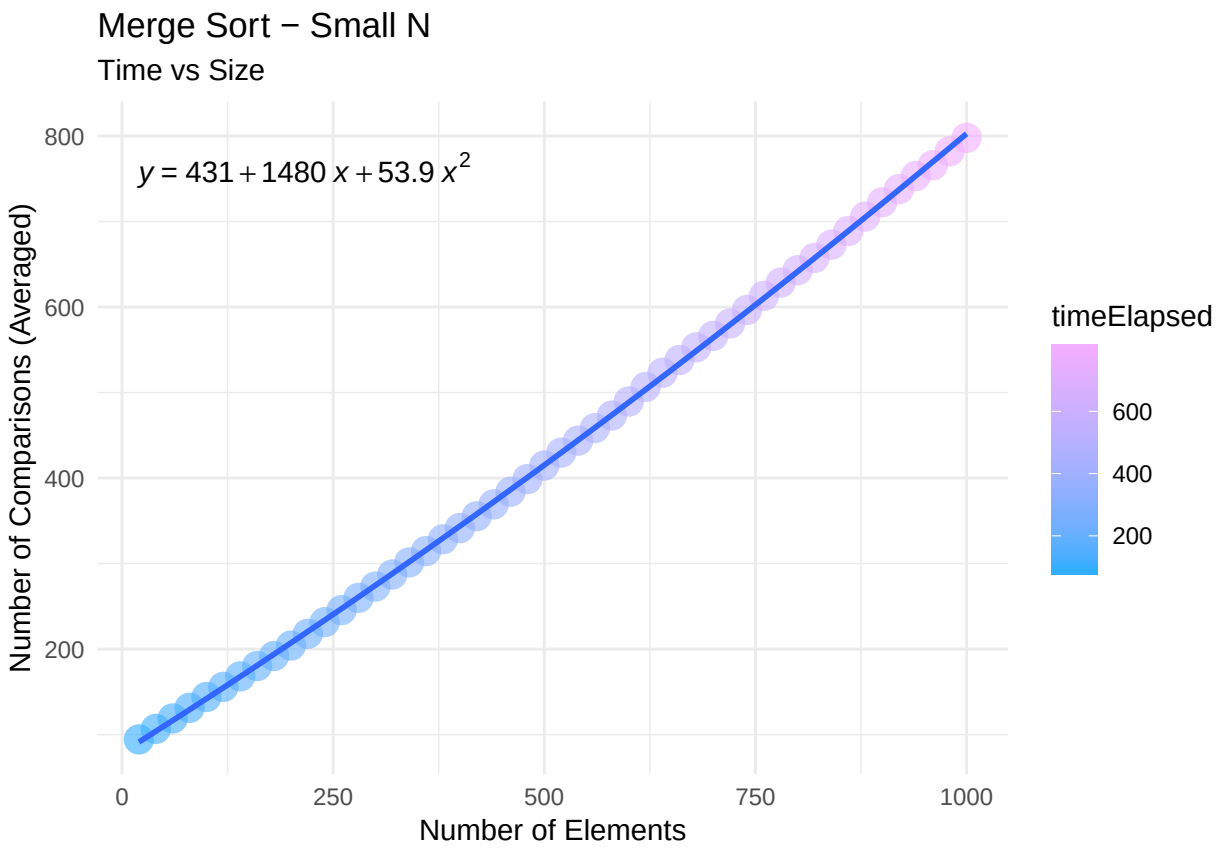
```
msdf_small <- replicator(mergeSort)
msdf_small
```

```
##     ele timeElapsed
## 1    20        94.4
## 2    40       106.7
## 3    60       118.9
## 4    80       131.4
## 5   100       144.0
## 6   120       155.8
## 7   140       168.1
## 8   160       180.2
## 9   180       191.9
## 10  200       204.1
## 11  220       217.9
## 12  240       231.5
## 13  260       245.6
## 14  280       259.9
## 15  300       273.1
## 16  320       287.2
## 17  340       301.4
```

```
## 18  360       314.6
## 19  380       328.4
## 20  400       341.5
## 21  420       355.4
## 22  440       369.2
## 23  460       384.0
## 24  480       398.8
## 25  500       414.7
## 26  520       429.7
## 27  540       443.8
## 28  560       458.4
## 29  580       473.1
## 30  600       489.5
## 31  620       506.5
## 32  640       523.1
## 33  660       538.0
## 34  680       552.9
## 35  700       566.3
## 36  720       580.8
## 37  740       596.6
## 38  760       613.0
## 39  780       628.5
## 40  800       643.0
## 41  820       657.5
## 42  840       672.9
## 43  860       688.9
## 44  880       705.8
## 45  900       722.5
## 46  920       737.9
## 47  940       753.2
## 48  960       765.8
## 49  980       782.1
## 50 1000       797.9
```

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

```
## Warning: Ignoring unknown parameters: rm
```



Quick Sort

Sorting Algorithm

```
quickSort <- function(vec, low = 1, high = length(vec)){

  partition <- function(vec, low, high){
    i = low
    op <- 0
    pivot = vec[high]
    for(j in low:(high - 1)){
      op <- op + 1
      if(vec[j] <= pivot){
        temp = vec[i]
        vec[i] = vec[j]
        vec[j] = temp
        i = i + 1
      }
    }
    temp = vec[i]
    vec[i] = vec[high]
    vec[high] = temp
    return (list("vec" = vec, "operations" = op, "pi" = i))
  }
```

```
  op <- 0
  if(low < high){
    pi_list = partition(vec, low, high)
    vec <- pi_list$vec
    pi <- pi_list$pi

    left_list <- quickSort(vec, low, pi - 1)
    vec <- left_list$vec

    right_list <- quickSort(vec, pi + 1, high)
    vec <- right_list$vec

    op <- op + left_list$operations + right_list$operations + pi_list$operations
    return (list("vec" = vec, "operations" = op))
  }else{
    return (list("vec" = vec, "operations" = op))
  }
}
```

Proof of Concept

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

```
## $vec
## [1] -33 -22   2   2  12  13
##
## $operations
## [1] 9
```

RunTime and Plot

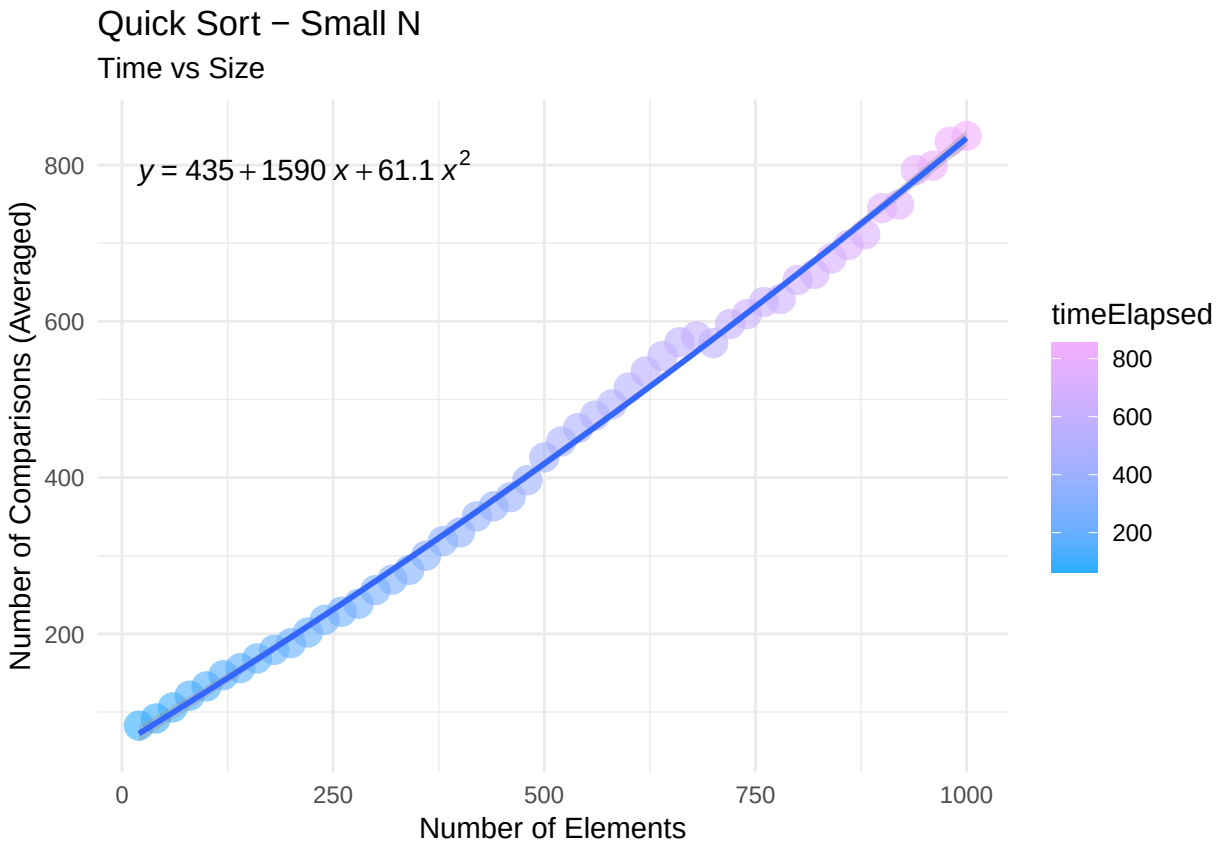
```
qsdf_small <- replicator(quickSort)
qsdf_small
```

```
##     ele timeElapsed
## 1    20        82.6
## 2    40        91.7
## 3    60       105.9
## 4    80       120.9
## 5   100       132.8
## 6   120       147.2
## 7   140       156.2
## 8   160       168.6
## 9   180       179.5
## 10  200       188.2
## 11  220       201.7
## 12  240       218.0
## 13  260       228.3
## 14  280       238.5
## 15  300       256.0
## 16  320       269.5
```

```
## 17  340       281.8
## 18  360       300.2
## 19  380       318.8
## 20  400       329.8
## 21  420       350.4
## 22  440       363.3
## 23  460       375.0
## 24  480       396.7
## 25  500       425.9
## 26  520       446.2
## 27  540       463.4
## 28  560       479.4
## 29  580       493.7
## 30  600       515.2
## 31  620       535.8
## 32  640       555.6
## 33  660       573.3
## 34  680       580.9
## 35  700       572.2
## 36  720       596.5
## 37  740       609.2
## 38  760       624.8
## 39  780       628.7
## 40  800       652.9
## 41  820       660.7
## 42  840       680.5
## 43  860       697.7
## 44  880       711.3
## 45  900       744.9
## 46  920       749.4
## 47  940       793.6
## 48  960       799.4
## 49  980       830.0
## 50 1000       837.3
```

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

```
## Warning: Ignoring unknown parameters: rm
```



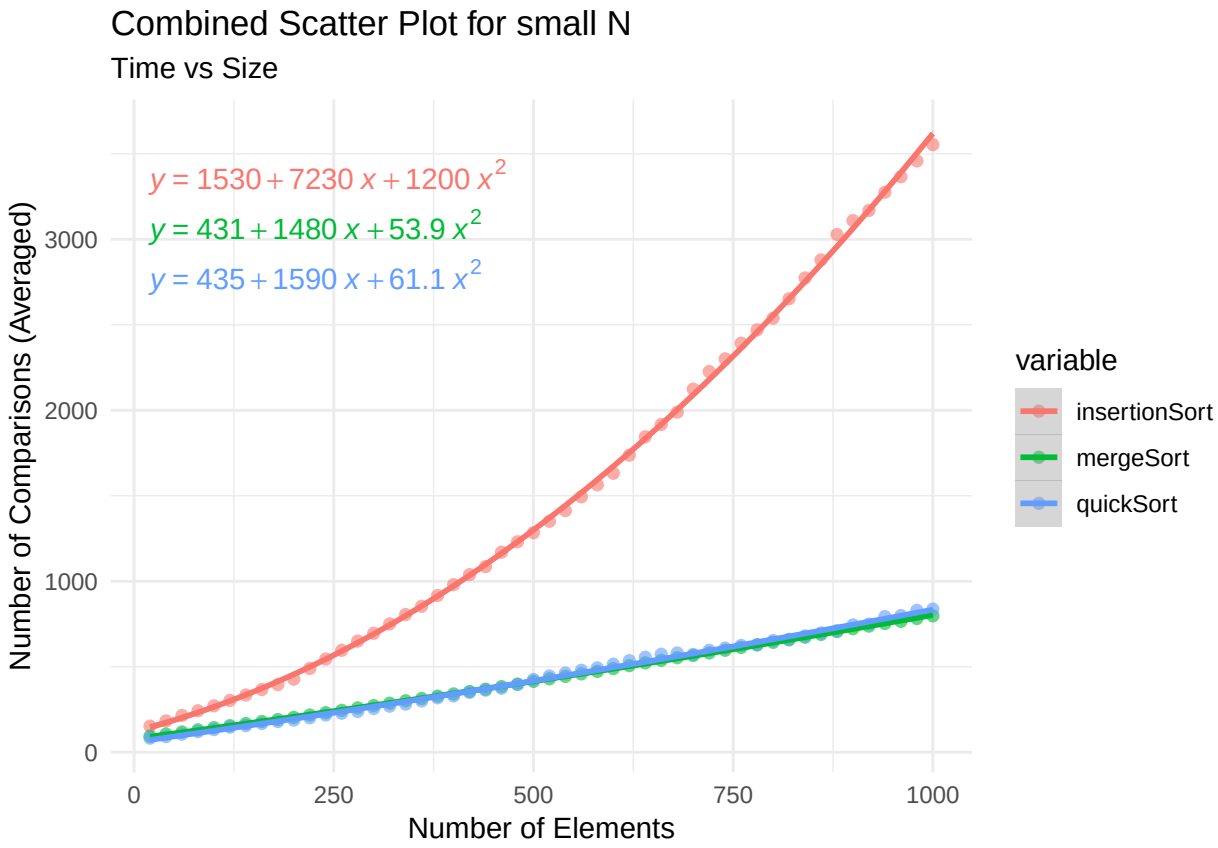
Combined Plots

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

```
##     ele insertionSort mergeSort quickSort
## 1    20         153.1      94.4      82.6
## 2    40         183.5     106.7      91.7
## 3    60         215.6     118.9     105.9
## 4    80         242.9     131.4     120.9
## 5   100         271.7     144.0     132.8
## 6   120         303.2     155.8     147.2
## 7   140         334.8     168.1     156.2
## 8   160         366.8     180.2     168.6
## 9   180         396.3     191.9     179.5
## 10  200         427.8     204.1     188.2
## 11  220         490.5     217.9     201.7
## 12  240         545.4     231.5     218.0
## 13  260         596.4     245.6     228.3
## 14  280         650.2     259.9     238.5
## 15  300         695.8     273.1     256.0
## 16  320         750.8     287.2     269.5
```

```
## 17  340         805.9     301.4     281.8
## 18  360         853.9     314.6     300.2
## 19  380         916.7     328.4     318.8
## 20  400         980.4     341.5     329.8
## 21  420        1039.0     355.4     350.4
## 22  440        1086.2     369.2     363.3
## 23  460        1170.2     384.0     375.0
## 24  480        1230.9     398.8     396.7
## 25  500        1284.2     414.7     425.9
## 26  520        1351.1     429.7     446.2
## 27  540        1413.4     443.8     463.4
## 28  560        1494.1     458.4     479.4
## 29  580        1563.9     473.1     493.7
## 30  600        1631.7     489.5     515.2
## 31  620        1737.4     506.5     535.8
## 32  640        1844.9     523.1     555.6
## 33  660        1917.0     538.0     573.3
## 34  680        1989.2     552.9     580.9
## 35  700        2123.6     566.3     572.2
## 36  720        2226.5     580.8     596.5
## 37  740        2300.9     596.6     609.2
## 38  760        2392.4     613.0     624.8
## 39  780        2470.2     628.5     628.7
## 40  800        2538.5     643.0     652.9
## 41  820        2652.2     657.5     660.7
## 42  840        2773.7     672.9     680.5
## 43  860        2879.5     688.9     697.7
## 44  880        3027.8     705.8     711.3
## 45  900        3109.4     722.5     744.9
## 46  920        3168.4     737.9     749.4
## 47  940        3275.0     753.2     793.6
## 48  960        3366.0     765.8     799.4
## 49  980        3458.7     782.1     830.0
## 50 1000        3553.5     797.9     837.3
```

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


Combined Scatter Plot for small N
Time vs Size
$y = 1530 + 7230\,x + 1200\,x^2$
$y = 431 + 1480\,x + 53.9\,x^2$
$y = 435 + 1590\,x + 61.1\,x^2$
Number of Comparisons (Averaged)
0
1000
2000
3000
0
250
500
750
1000
Number of Elements
variable
insertionSort
mergeSort
quickSort