SortTimes

Samyak Ahuja

August 23, 2018

Complexity for different Sorting Algorithms.

Helper Functions

Replicator

```
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(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

```
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
```

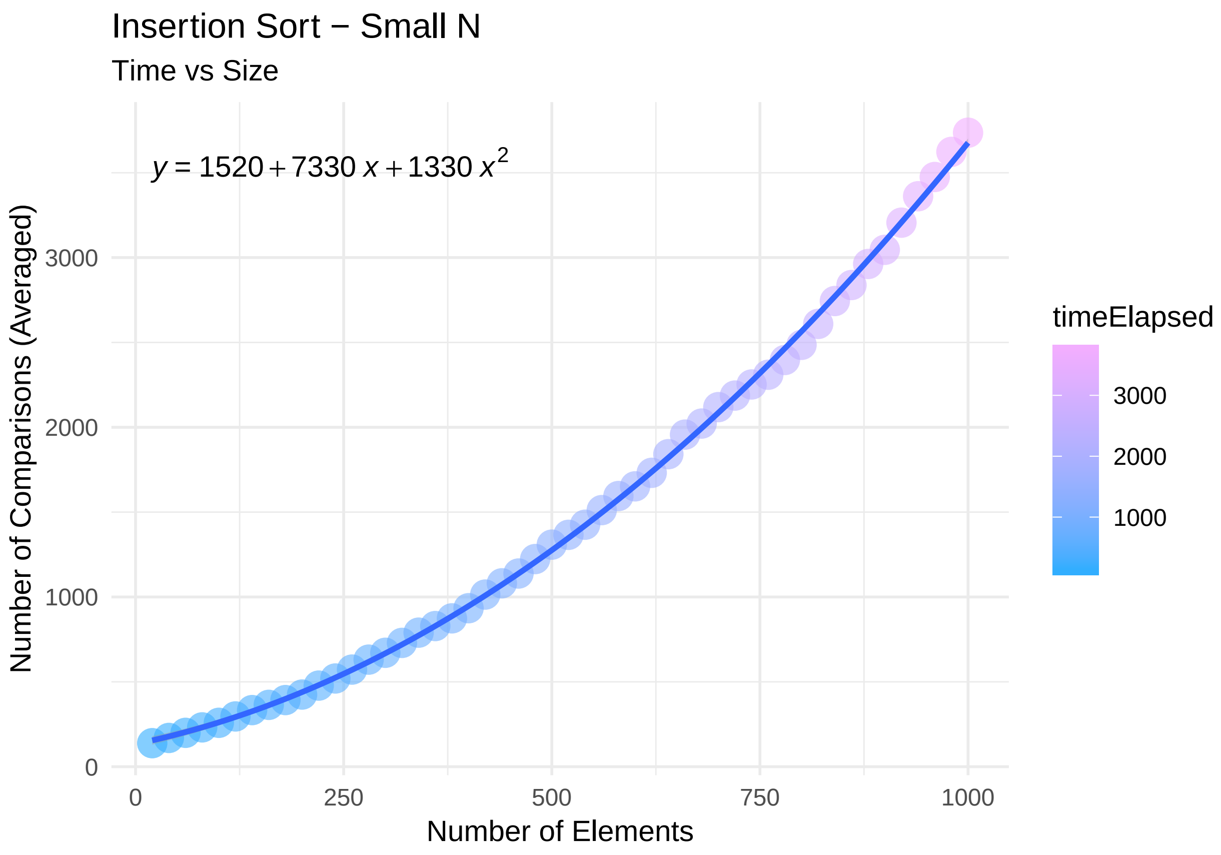
RunTime and Plot

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

```
##     ele timeElapsed
## 1    20       138.5
## 2    40       169.3
## 3    60       199.4
## 4    80       231.6
## 5   100       258.5
## 6   120       296.0
## 7   140       333.4
## 8   160       364.3
## 9   180       392.5
## 10  200       425.0
## 11  220       477.8
## 12  240       519.7
## 13  260       572.1
## 14  280       630.8
## 15  300       671.0
## 16  320       729.4
## 17  340       789.8
## 18  360       828.7
## 19  380       874.1
## 20  400       933.8
## 21  420      1014.2
## 22  440      1080.6
## 23  460      1138.2
## 24  480      1223.5
## 25  500      1308.4
## 26  520      1366.3
## 27  540      1425.8
## 28  560      1512.0
## 29  580      1594.9
## 30  600      1652.0
## 31  620      1731.7
## 32  640      1841.6
## 33  660      1956.8
## 34  680      2022.0
## 35  700      2118.8
## 36  720      2186.9
## 37  740      2252.3
## 38  760      2309.7
## 39  780      2396.6
## 40  800      2485.9
## 41  820      2609.3
## 42  840       2744.1
## 43  860       2838.4
## 44  880       2962.5
## 45  900       3045.8
## 46  920       3205.9
## 47  940       3361.6
## 48  960       3474.9
## 49  980       3623.0
## 50 1000       3736.0
```

```
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]){
```

```r
        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

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

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

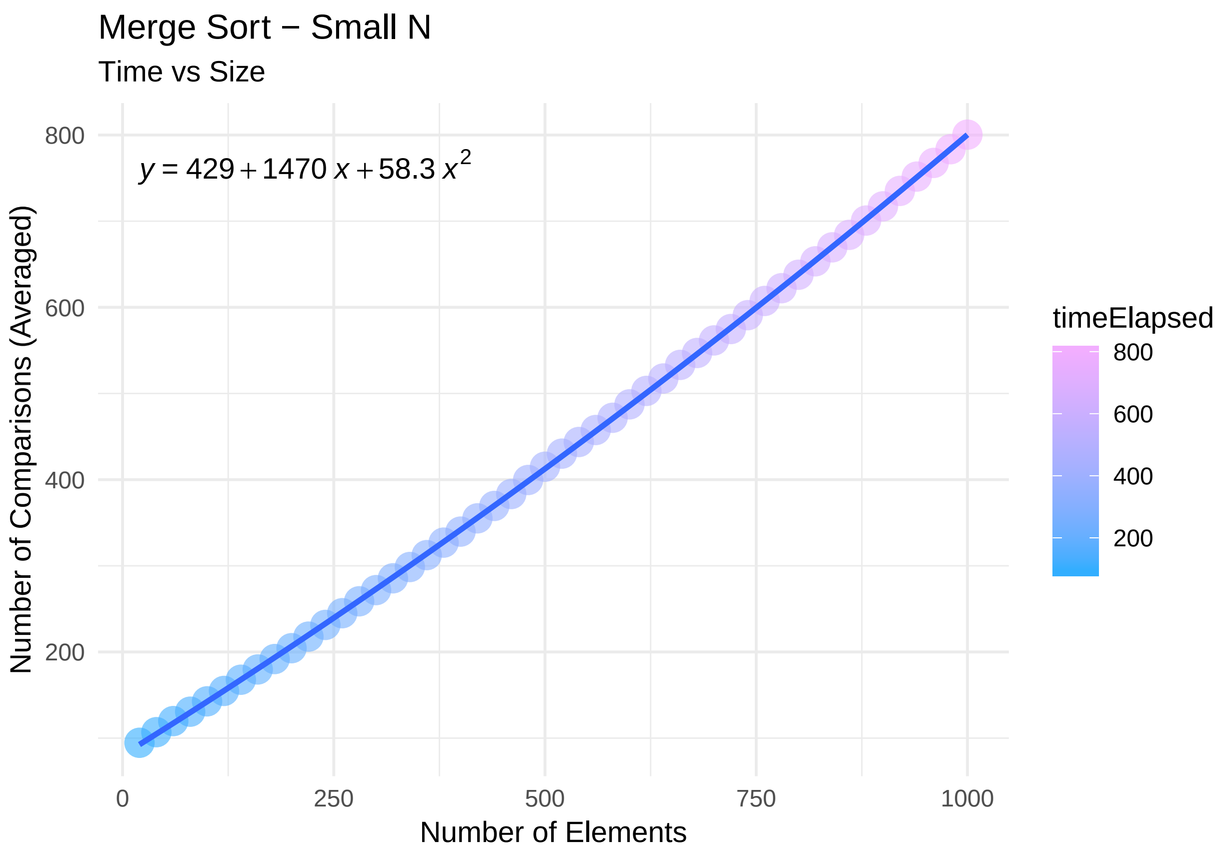
RunTime and Plot

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

```
##     ele timeElapsed
## 1    20        94.6
## 2    40       106.9
## 3    60       119.8
## 4    80        130.7
## 5   100        142.7
## 6   120        154.8
## 7   140        167.8
## 8   160        179.7
## 9   180        191.8
## 10  200        204.4
## 11  220        217.8
## 12  240        231.4
## 13  260        245.0
## 14  280        258.8
## 15  300        271.8
## 16  320        285.5
## 17  340        298.6
## 18  360        312.4
## 19  380        326.9
## 20  400        339.7
## 21  420        355.1
## 22  440        369.5
## 23  460        383.3
## 24  480        399.6
## 25  500        415.0
## 26  520        430.2
## 27  540        443.7
## 28  560        457.6
## 29  580        471.8
## 30  600        487.4
## 31  620        503.0
## 32  640        517.4
## 33  660        533.2
## 34  680        547.0
## 35  700        561.6
## 36  720        574.8
## 37  740        590.9
## 38  760        607.4
## 39  780        622.2
## 40  800        637.8
## 41  820        653.4
## 42  840        669.6
## 43  860        684.1
## 44  880        700.7
## 45  900        717.2
## 46  920        735.1
## 47  940        751.7
## 48  960        767.6
## 49  980        783.5
## 50 1000        800.4
```

```r
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
```

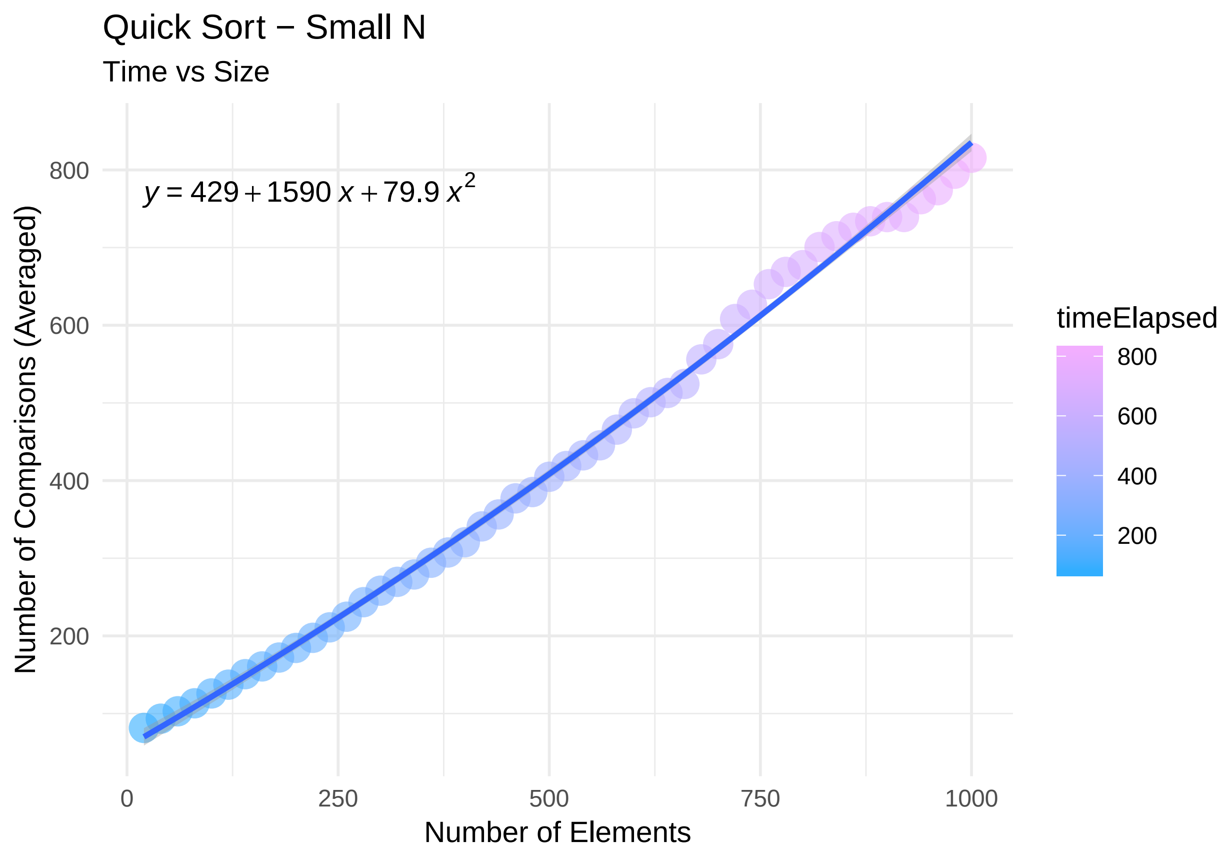
RunTime and Plot

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

```
##     ele timeElapsed
## 1    20        81.5
## 2    40        93.4
## 3    60       102.9
## 4    80       113.2
## 5   100       125.9
## 6   120       137.2
## 7   140       150.7
## 8   160       160.7
## 9   180       172.1
## 10  200       184.5
## 11  220       197.5
## 12  240       211.0
## 13  260       224.7
## 14  280       243.4
## 15  300       258.2
## 16  320       269.6
## 17  340       279.1
## 18  360       294.2
## 19  380       307.4
## 20  400       320.6
## 21  420       341.0
## 22  440       356.4
## 23  460       377.1
## 24  480       385.2
## 25  500       404.9
## 26  520       418.7
## 27  540       432.5
## 28  560       445.4
## 29  580       465.6
## 30  600       486.6
## 31  620       500.9
## 32  640       512.8
## 33  660       524.4
## 34  680       556.1
## 35  700       575.7
## 36  720       608.0
## 37  740       626.4
## 38  760       652.9
## 39  780       668.7
## 40  800       677.4
## 41  820       700.7
## 42  840       714.6
## 43  860       725.5
## 44  880       733.9
## 45  900       739.3
## 46  920       739.4
## 47  940       762.3
## 48  960       773.9
## 49  980       795.1
## 50 1000       815.7
```

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

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

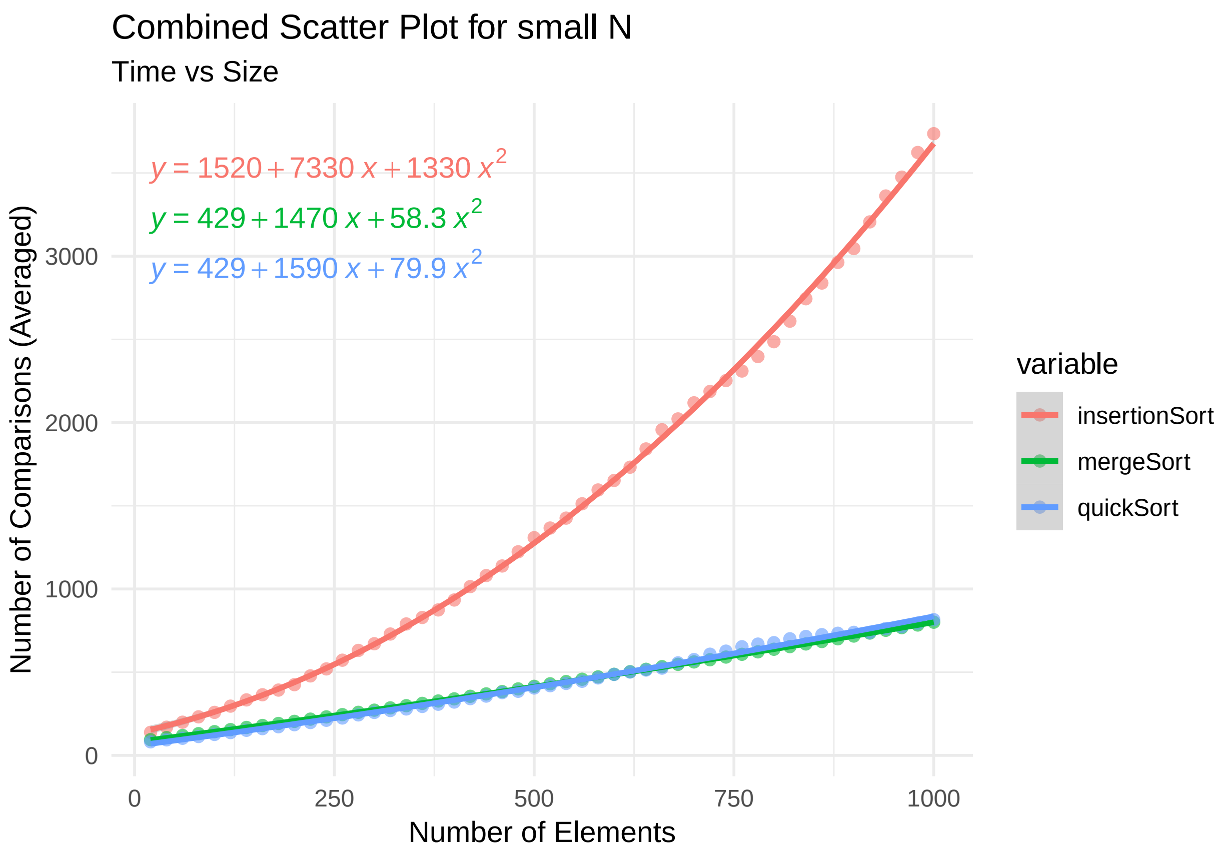


Combined Plots

```
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         138.5      94.6      81.5
## 2    40         169.3     106.9      93.4
## 3    60         199.4     119.8     102.9
## 4    80         231.6     130.7     113.2
## 5   100         258.5     142.7     125.9
## 6   120         296.0     154.8     137.2
## 7   140         333.4     167.8     150.7
## 8   160         364.3     179.7     160.7
## 9   180         392.5     191.8     172.1
## 10  200         425.0     204.4     184.5
## 11  220         477.8     217.8     197.5
## 12  240         519.7     231.4     211.0
## 13  260         572.1     245.0     224.7
## 14  280         630.8     258.8     243.4
## 15  300         671.0     271.8     258.2
## 16  320         729.4     285.5     269.6
## 17  340         789.8     298.6     279.1
## 18  360         828.7     312.4     294.2
## 19  380         874.1     326.9     307.4
## 20  400         933.8     339.7     320.6
## 21  420        1014.2     355.1     341.0
## 22  440        1080.6     369.5     356.4
## 23  460        1138.2     383.3     377.1
## 24  480        1223.5     399.6     385.2
## 25  500        1308.4     415.0     404.9
## 26  520        1366.3     430.2     418.7
## 27  540        1425.8     443.7     432.5
## 28  560        1512.0     457.6     445.4
## 29  580        1594.9     471.8     465.6
## 30  600        1652.0     487.4     486.6
## 31  620        1731.7     503.0     500.9
## 32  640        1841.6     517.4     512.8
## 33  660        1956.8     533.2     524.4
## 34  680        2022.0     547.0     556.1
## 35  700        2118.8     561.6     575.7
## 36  720        2186.9     574.8     608.0
## 37  740        2252.3     590.9     626.4
## 38  760        2309.7     607.4     652.9
## 39  780        2396.6     622.2     668.7
## 40  800        2485.9     637.8     677.4
## 41  820        2609.3     653.4     700.7
## 42  840        2744.1     669.6     714.6
## 43  860        2838.4     684.1     725.5
## 44  880        2962.5     700.7     733.9
## 45  900        3045.8     717.2     739.3
## 46  920        3205.9     735.1     739.4
## 47  940        3361.6     751.7     762.3
## 48  960        3474.9     767.6     773.9
## 49  980        3623.0     783.5     795.1
## 50 1000        3736.0     800.4     815.7
```

```
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
$y = 1520 + 7330\,x + 1330\,x^2$
$y = 429 + 1470\,x + 58.3\,x^2$
$y = 429 + 1590\,x + 79.9\,x^2$
Number of Comparisons (Averaged)
0
1000
2000
3000
0
250
500
750
1000
Number of Elements
variable
insertionSort
mergeSort
quickSort