Program 2 Graph Analysis

Ryan Schaefer and Wes Anderson

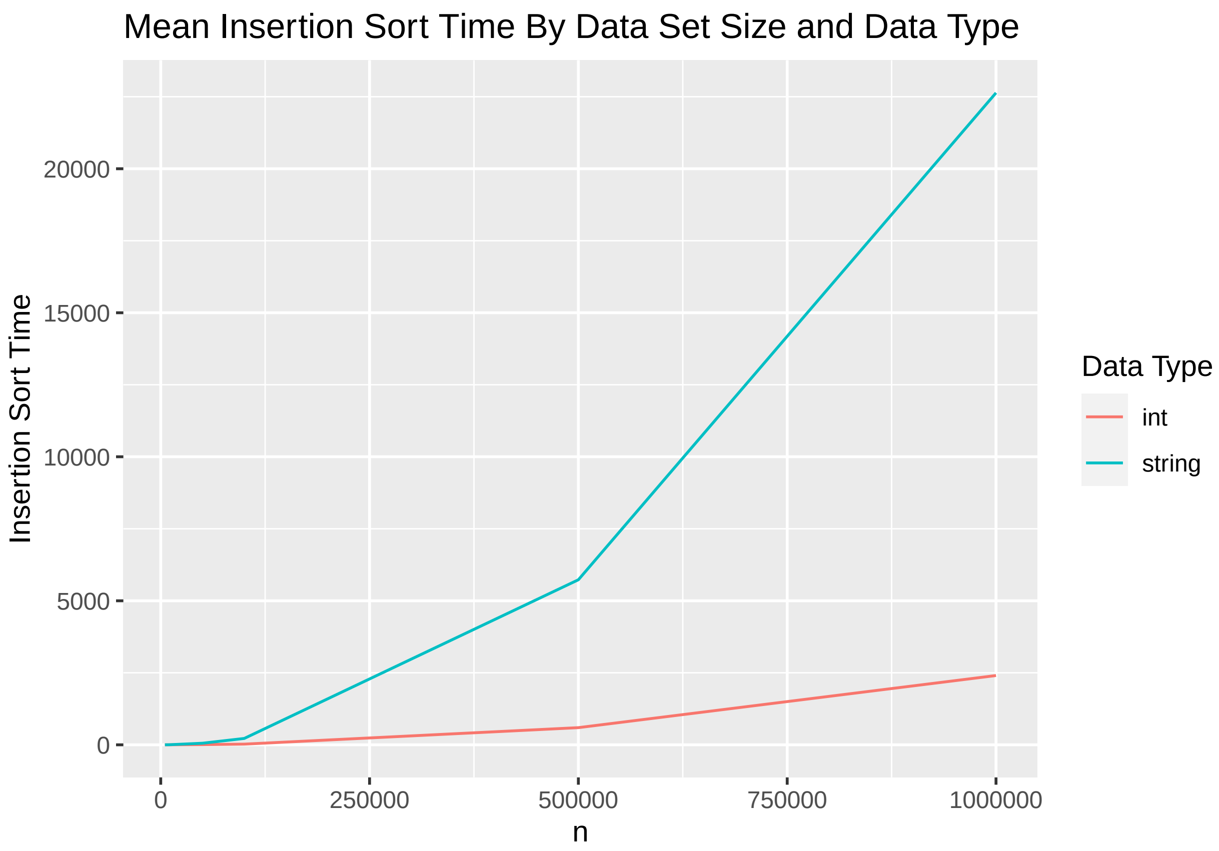
Create Dataset

```r
library(ggplot2)
library(ggpubr)
data = read.csv("CombinedMean.csv")
data$n2 = data$size ^ 2
data$nlogn = log(data$size) * data$size
data
```

```
##    var_type    size       format insertion_time  quick_time merge_time
## 1       int  500000 noDuplicates   9.285925e+02 0.206327000 2.61463833
## 2       int 1000000 40duplicates   3.794060e+03 0.382623167 5.22443000
## 3       int  100000 40duplicates   3.781830e+01 0.033999600 0.51216583
## 4       int   10000 40duplicates   3.922030e-01 0.003015658 0.04998903
## 5       int   50000       sorted   8.808050e-04 0.013769020 0.25437033
## 6       int   50000 20duplicates   9.505890e+00 0.016747400 0.24884150
## 7       int    5000 noDuplicates   9.755405e-02 0.001479500 0.02301990
## 8       int  500000       sorted   8.602980e-03 0.126128717 2.68396833
## 9       int  500000     60sorted   1.526105e+02 0.161700333 2.75717000
## 10      int   10000     60sorted   6.146205e-02 0.002638110 0.05102268
## 11      int 1000000 noDuplicates   3.801070e+03 0.411008167 5.63687500
## 12      int 1000000 20duplicates   3.821290e+03 0.414956833 5.54547500
## 13      int   50000 noDuplicates   9.588935e+00 0.016368800 0.24949867
## 14      int    5000     60sorted   1.570460e-02 0.001290430 0.02320570
## 15      int    5000       sorted   7.994000e-05 0.001094495 0.02176585
## 16      int  100000 20duplicates   3.806845e+01 0.034022883 0.50394017
## 17      int   50000     60sorted   1.527910e+00 0.013458817 0.23827350
## 18      int   10000 noDuplicates   3.878130e-01 0.003147355 0.04840523
## 19      int  500000 20duplicates   9.515705e+02 0.188005667 2.54086500
## 20      int  500000 40duplicates   9.473160e+02 0.187674167 2.51964500
## 21      int 1000000       sorted   1.718955e-02 0.268821000 5.03317167
## 22      int    5000 20duplicates   9.281250e-02 0.001463062 0.02251170
## 23      int  100000 noDuplicates   3.781050e+01 0.033718167 0.49470783
## 24      int   50000 40duplicates   9.435210e+00 0.018693150 0.24369433
## 25      int   10000 20duplicates   3.831360e-01 0.003024307 0.04613460
## 26      int  100000       sorted   1.724845e-03 0.023970783 0.48152800
## 27      int  100000     60sorted   6.081445e+00 0.027807483 0.48339017
## 28      int   10000       sorted   1.741500e-04 0.002271060 0.04452613
## 29      int    5000 40duplicates   9.843835e-02 0.001470613 0.02261000
## 30      int 1000000     60sorted   6.057955e+02 0.329097500 5.26134500
## 31   string   50000       sorted   7.071545e-03 0.102598900 0.35941650
## 32   string  500000 20duplicates   7.677610e+03 1.300619833 4.57768667
## 33   string   50000 20duplicates   7.631110e+01 0.102368983 0.42131017
## 34   string   10000 40duplicates   3.060785e+00 0.017345450 0.07621432
## 35   string   10000     60sorted   1.941865e+00 0.017530300 0.07196508
## 36   string  100000       sorted   1.481790e-02 0.207820333 0.75133783
## 37   string    5000 40duplicates   7.553355e-01 0.008053643 0.03603047
## 38   string  500000     60sorted   4.921090e+03 1.244085333 4.07648833
## 39   string   50000 noDuplicates   7.389455e+01 0.102702433 0.40604750
## 40   string  500000 40duplicates   7.515225e+03 1.331646667 4.44772833
## 41   string    5000 20duplicates   7.489175e-01 0.008284670 0.03635257
## 42   string  100000 noDuplicates   3.005005e+02 0.221395667 0.84280300
## 43   string    5000 noDuplicates   7.578600e-01 0.008016898 0.03610372
## 44   string  100000     60sorted   1.924050e+02 0.215852667 0.78265900
## 45   string 1000000 20duplicates   3.054190e+04 2.752548333 9.25050000
## 46   string   10000 noDuplicates   3.038025e+00 0.020355300 0.07655670
## 47   string 1000000 noDuplicates   3.192455e+04 2.953643333 9.94392833
## 48   string 1000000       sorted   1.652630e-01 2.693163333 8.97265333
## 49   string  500000 noDuplicates   8.534840e+03 1.290818333 4.45609500
## 50   string  100000 40duplicates   3.049645e+02 0.230166000 0.84875567
## 51   string    5000     60sorted   4.765915e-01 0.007937330 0.03518605
## 52   string 1000000     60sorted   1.995215e+04 2.674513333 8.52168500
## 53   string    5000       sorted   6.632210e-04 0.007405297 0.03308105
## 54   string  100000 20duplicates   3.070495e+02 0.223292833 0.84737950
## 55   string   10000 20duplicates   3.045350e+00 0.019640950 0.07947848
## 56   string   10000       sorted   1.413440e-03 0.016957600 0.06788902
## 57   string  500000       sorted   7.140805e-02 1.230575333 3.93005833
## 58   string   50000 40duplicates   7.663050e+01 0.106078450 0.41228883
## 59   string   50000     60sorted   4.958145e+01 0.103647050 0.37648733
## 60   string 1000000 40duplicates   3.075595e+04 2.695923333 9.21054500
##     shell_time  intro_time    tim_time      n2       nlogn
## 1  0.352971000 0.843892000 0.947042667 2.5e+11  6561181.69
## 2  0.776073667 1.920311667 2.012676667 1.0e+12 13815510.56
## 3  0.056593233 0.167453833 0.184260667 1.0e+10  1151292.55
## 4  0.003537902 0.013601550 0.015257670 1.0e+08    92103.40
## 5  0.006851010 0.070738550 0.067643983 2.5e+09   540988.91
## 6  0.023602950 0.075824083 0.082080167 2.5e+09   540988.91
## 7  0.001568615 0.005962470 0.006346413 2.5e+07    42585.97
## 8  0.093804217 0.947332333 0.913185833 2.5e+11  6561181.69
## 9  0.183444000 0.926444333 0.946440667 2.5e+11  6561181.69
## 10 0.002130168 0.014277322 0.014156652 1.0e+08    92103.40
## 11 0.812542667 1.955400000 2.130015000 1.0e+12 13815510.56
## 12 0.798510333 1.929925000 2.034711667 1.0e+12 13815510.56
## 13 0.024548200 0.075524067 0.081699667 2.5e+09   540988.91
## 14 0.000932098 0.005896533 0.005888852 2.5e+07    42585.97
## 15 0.000519300 0.005489445 0.005415458 2.5e+07    42585.97
## 16 0.052829050 0.156468333 0.174708500 1.0e+10  1151292.55
## 17 0.013072938 0.070147767 0.070103217 2.5e+09   540988.91
## 18 0.003787818 0.013667925 0.015091280 1.0e+08    92103.40
## 19 0.352763500 0.874192167 0.947142500 2.5e+11  6561181.69
## 20 0.342163333 0.867710333 0.932384000 2.5e+11  6561181.69
## 21 0.182980500 1.726065000 1.692198333 1.0e+12 13815510.56
## 22 0.001509243 0.005781012 0.006353145 2.5e+07    42585.97
## 23 0.054667500 0.156110833 0.168874333 1.0e+10  1151292.55
## 24 0.023900867 0.075078100 0.078798717 2.5e+09   540988.91
## 25 0.003535478 0.012398400 0.014153950 1.0e+08    92103.40
## 26 0.015084163 0.145525500 0.144335667 1.0e+10  1151292.55
## 27 0.029143467 0.153611167 0.149616667 1.0e+10  1151292.55
## 28 0.001164397 0.012083587 0.011516255 1.0e+08    92103.40
## 29 0.001644188 0.006400853 0.007144227 2.5e+07    42585.97
## 30 0.381220500 1.815090000 1.829731667 1.0e+12 13815510.56
## 31 0.077971767 0.196735167 0.177451667 2.5e+09   540988.91
## 32 3.145688333 2.708421667 3.201145000 2.5e+11  6561181.69
## 33 0.211183500 0.219723000 0.261063167 2.5e+09   540988.91
## 34 0.029625367 0.035755483 0.045282100 1.0e+08    92103.40
## 35 0.030016500 0.036678100 0.036267417 1.0e+08    92103.40
## 36 0.173364667 0.418632500 0.384484667 1.0e+10  1151292.55
## 37 0.012812505 0.016618183 0.021617067 2.5e+07    42585.97
## 38 2.780971667 2.537850000 2.576486667 2.5e+11  6561181.69
## 39 0.213212167 0.208330833 0.258270000 2.5e+09   540988.91
## 40 3.353028333 2.691278333 3.154970000 2.5e+11  6561181.69
## 41 0.012585220 0.017345617 0.019759567 2.5e+07    42585.97
## 42 0.474979667 0.453595667 0.555115833 1.0e+10  1151292.55
## 43 0.013104647 0.016824250 0.019876750 2.5e+07    42585.97
## 44 0.454941333 0.439172500 0.455344833 1.0e+10  1151292.55
## 45 7.587153333 5.670558333 6.678626667 1.0e+12 13815510.56
## 46 0.031396567 0.043000917 0.045159117 1.0e+08    92103.40
## 47 7.754373333 6.186868333 7.270081667 1.0e+12 13815510.56
## 48 2.071555000 6.269931667 5.111798333 1.0e+12 13815510.56
## 49 3.092020000 2.599186667 3.212418333 2.5e+11  6561181.69
## 50 0.488783333 0.458252333 0.556232000 1.0e+10  1151292.55
## 51 0.011447625 0.015932483 0.016620467 2.5e+07    42585.97
## 52 6.286810000 5.609093333 5.567153333 1.0e+12 13815510.56
## 53 0.006246722 0.016007283 0.013668128 2.5e+07    42585.97
## 54 0.499944667 0.490365167 0.569593000 1.0e+10  1151292.55
## 55 0.029829383 0.034127300 0.043808050 1.0e+08    92103.40
## 56 0.013447902 0.034099067 0.032859733 1.0e+08    92103.40
## 57 0.963687500 2.504511667 2.234748333 2.5e+11  6561181.69
## 58 0.213883500 0.213099000 0.260947500 2.5e+09   540988.91
## 59 0.199950000 0.211369833 0.213792333 2.5e+09   540988.91
## 60 7.224503333 5.739863333 6.769823333 1.0e+12 13815510.56
```

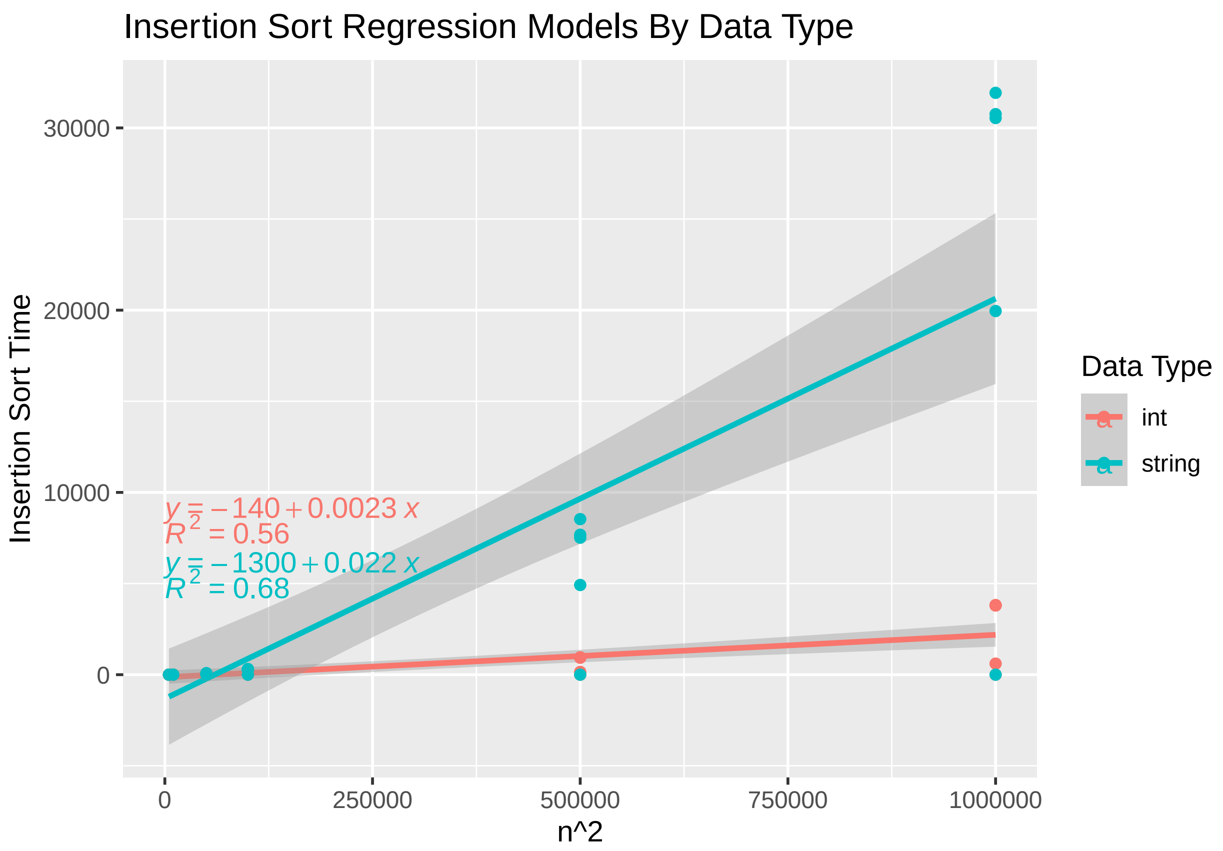
Insertion Sort

```
insertionTimes = aggregate(insertion_time ~ var_type + size + n2 + format, data = data, FUN = mean)
insertionTimes2 = aggregate(insertion_time ~ var_type + size + n2, data = data, FUN = mean)
ggplot(insertionTimes2, aes(x = size, y = insertion_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Insertion Sort Time By Data Set Size and Data Type", x = "n", y = "Insertion Sort T
  guides(color = guide_legend(title = "Data Type"))
```

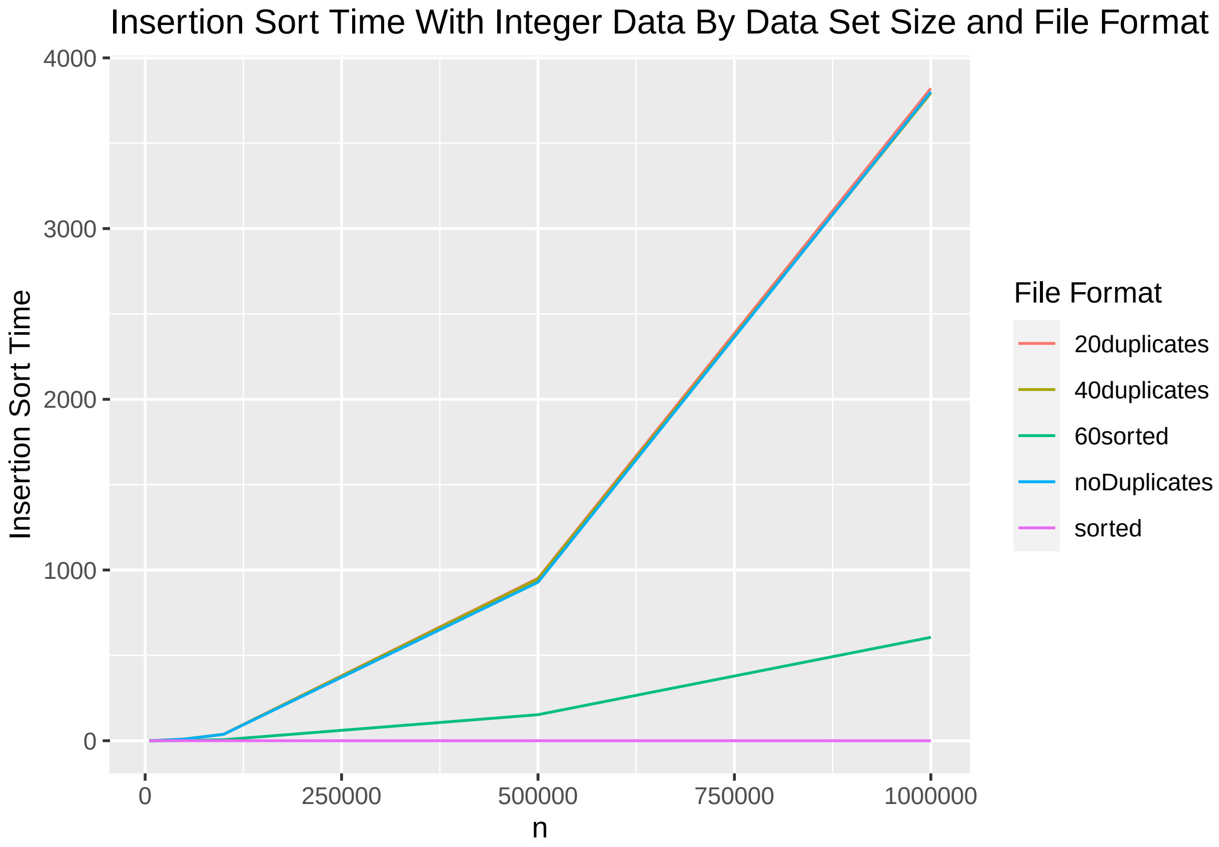


```
ggplot(insertionTimes, aes(x = size, y = insertion_time, color = var_type)) +
  labs(title = "Insertion Sort Regression Models By Data Type", x = "n^2", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(9000, 6000)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(8000, 5000)) +
  guides(color = guide_legend(title = "Data Type"))
```

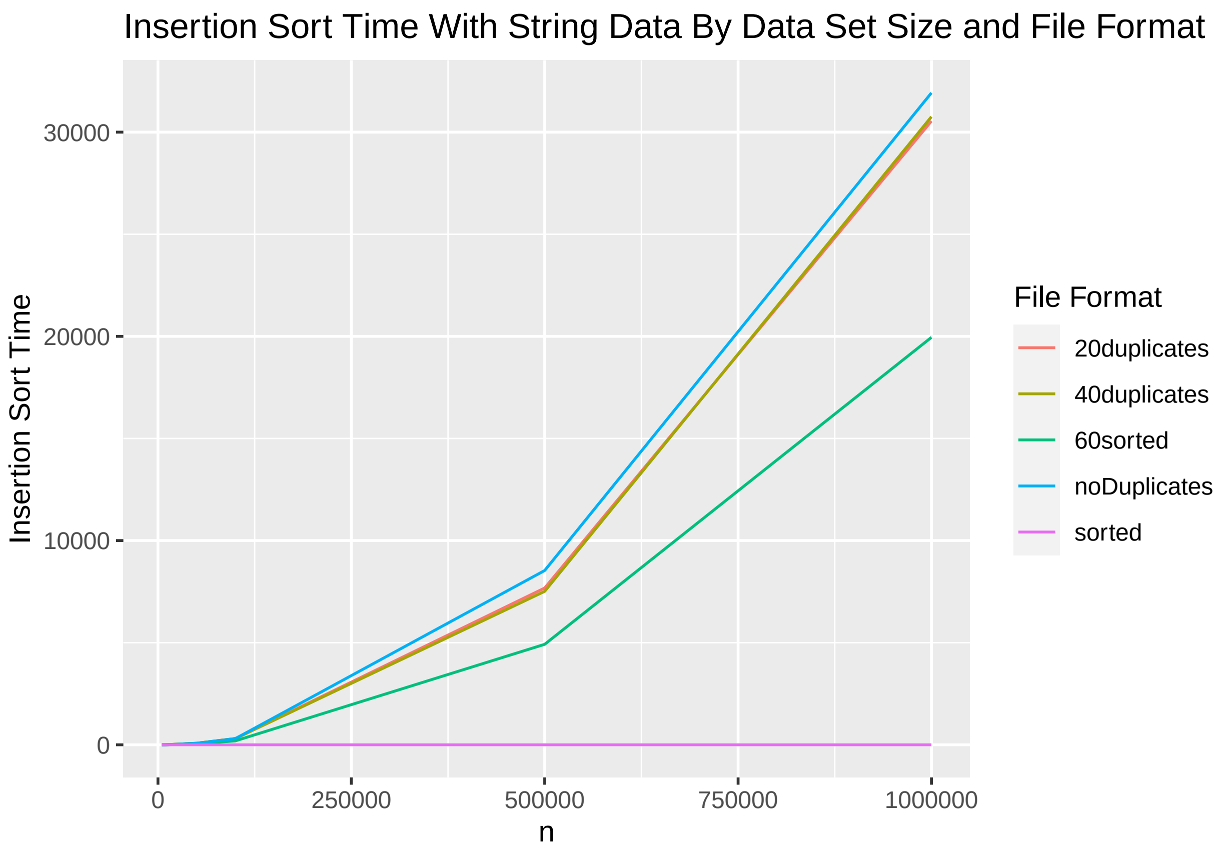
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
insertionInts = subset(insertionTimes, var_type == "int")
ggplot(insertionInts, aes(x = size, y = insertion_time, color = format)) +
  geom_line() +
  labs(title = "Insertion Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "
  guides(color = guide_legend(title = "File Format"))
```

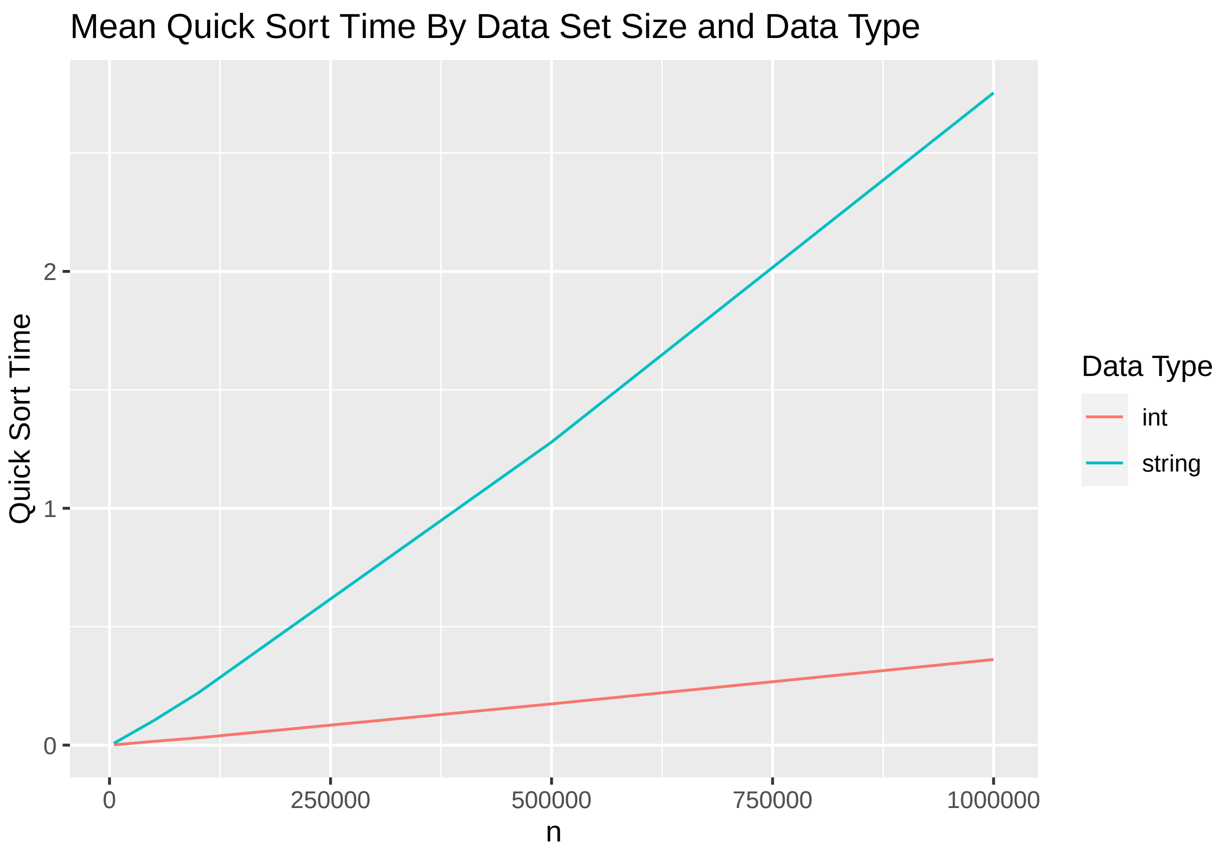


```r
insertionStrings = subset(insertionTimes, var_type == "string")
ggplot(insertionStrings, aes(x = size, y = insertion_time, color = format)) +
  geom_line() +
  labs(title = "Insertion Sort Time With String Data By Data Set Size and File Format", x = "n", y = "In
  guides(color = guide_legend(title = "File Format"))
```



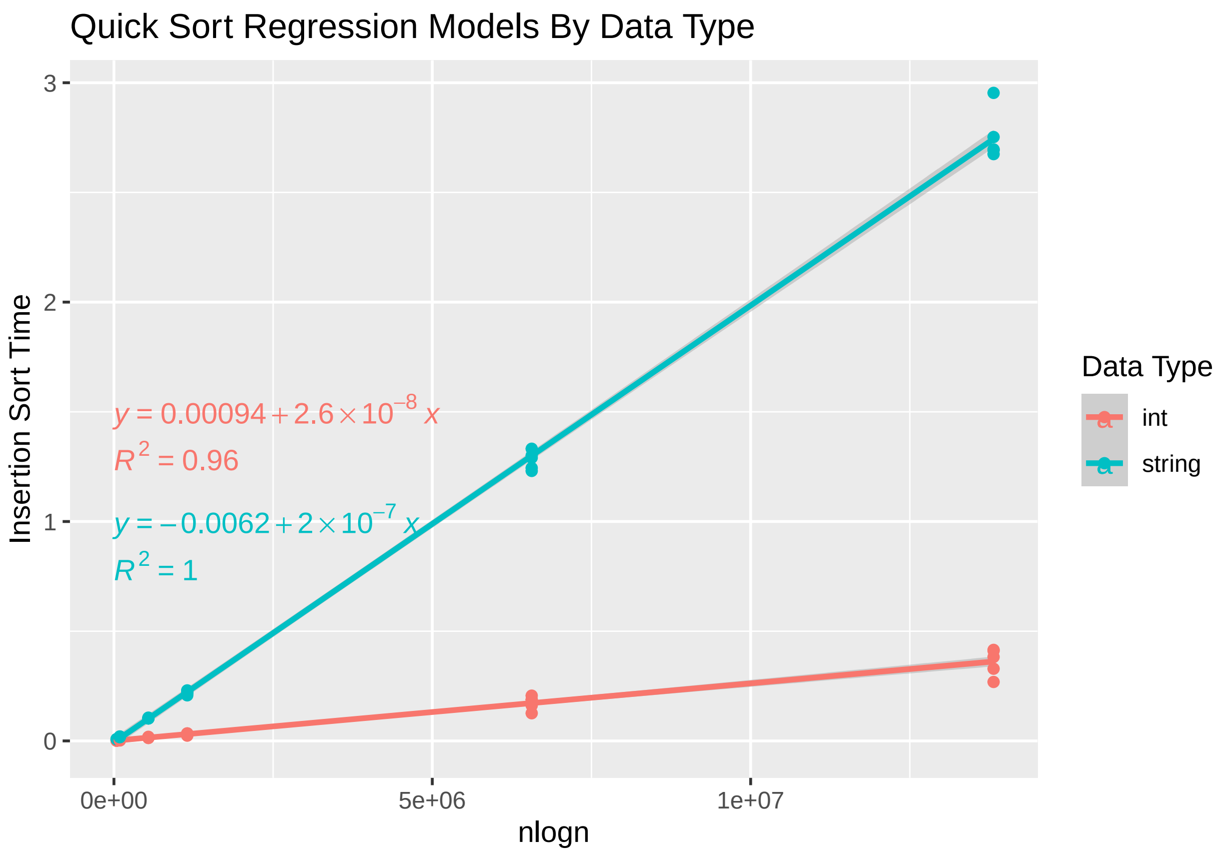
Quick Sort

```
quickTimes = aggregate(quick_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
quickTimes2 = aggregate(quick_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(quickTimes2, aes(x = size, y = quick_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Quick Sort Time By Data Set Size and Data Type", x = "n", y = "Quick Sort Time") +
  guides(color = guide_legend(title = "Data Type"))
```

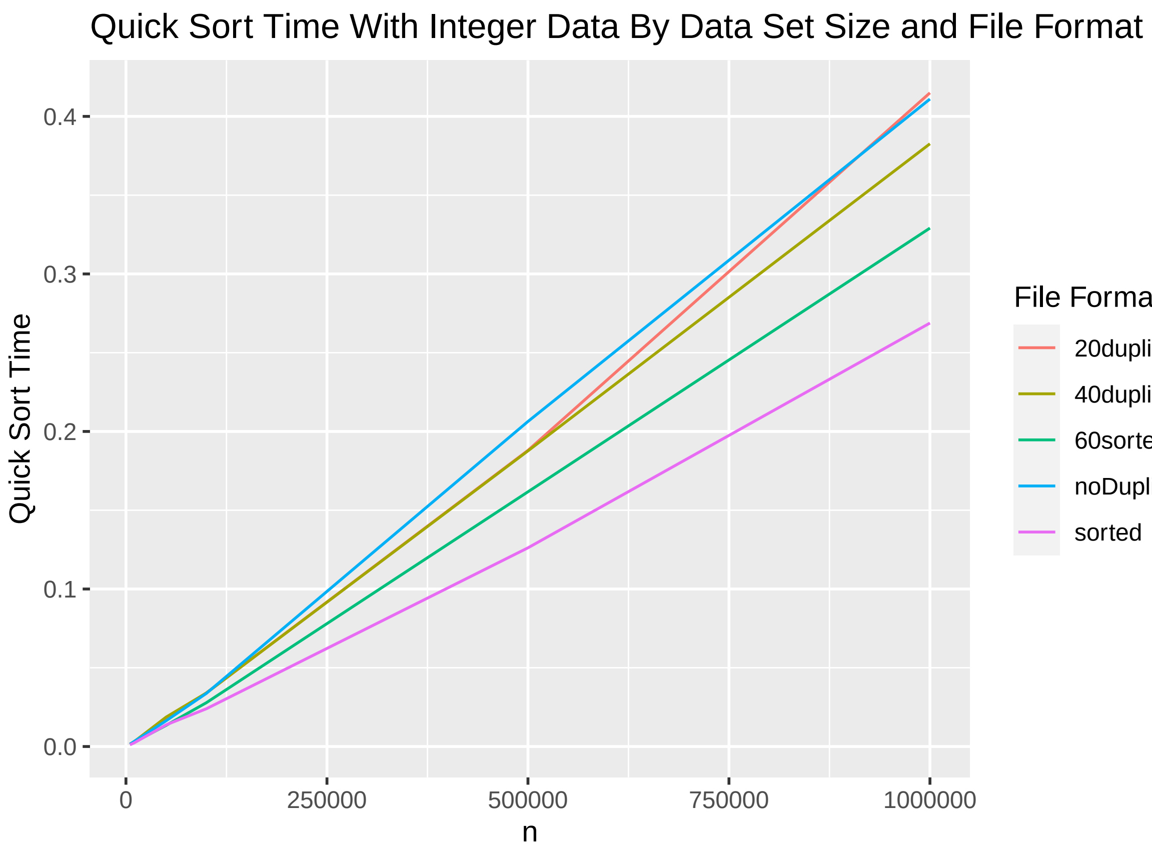


```
ggplot(quickTimes, aes(x = nlogn, y = quick_time, color = var_type)) +
  labs(title = "Quick Sort Regression Models By Data Type", x = "nlogn", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(1.5, 1)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(1.3, 0.8)) +
  guides(color = guide_legend(title = "Data Type"))
```

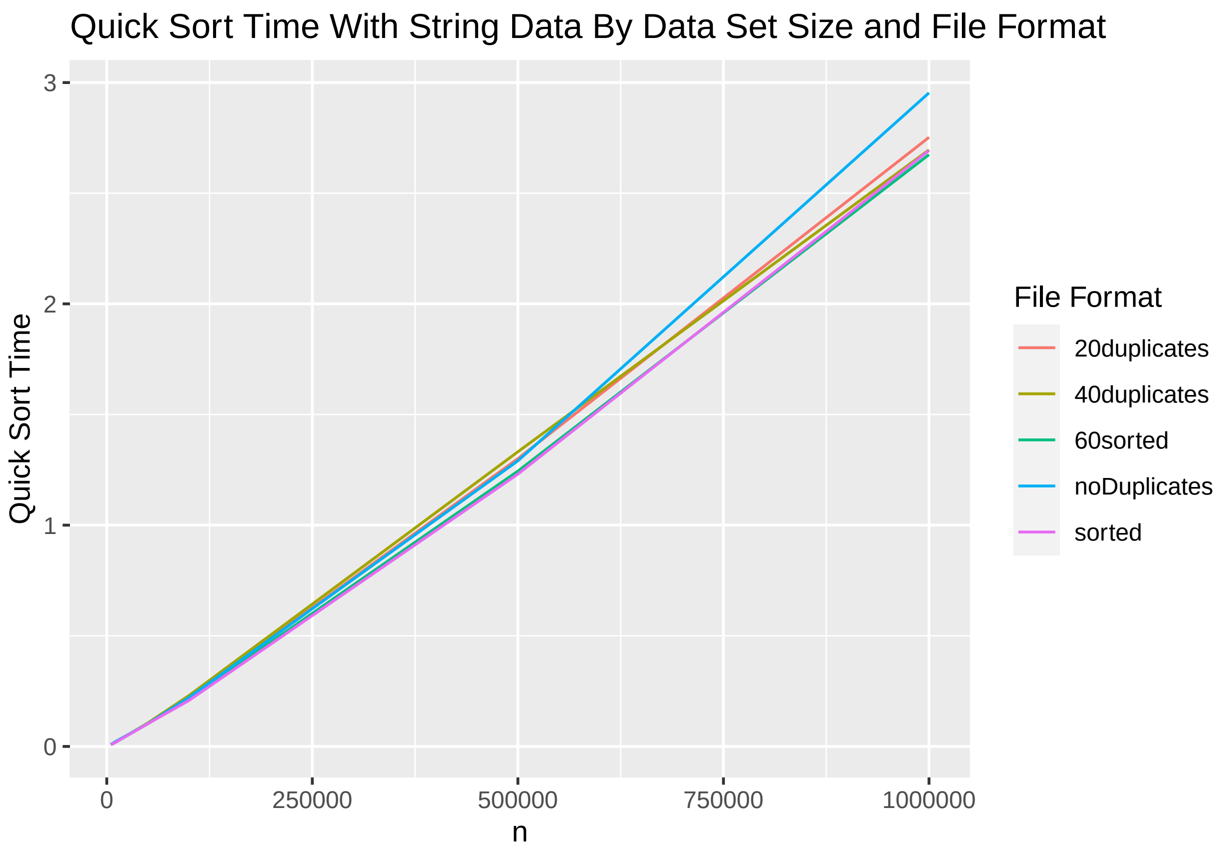
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
quickInts = subset(quickTimes, var_type == "int")
ggplot(quickInts, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Quic
  guides(color = guide_legend(title = "File Format"))
```

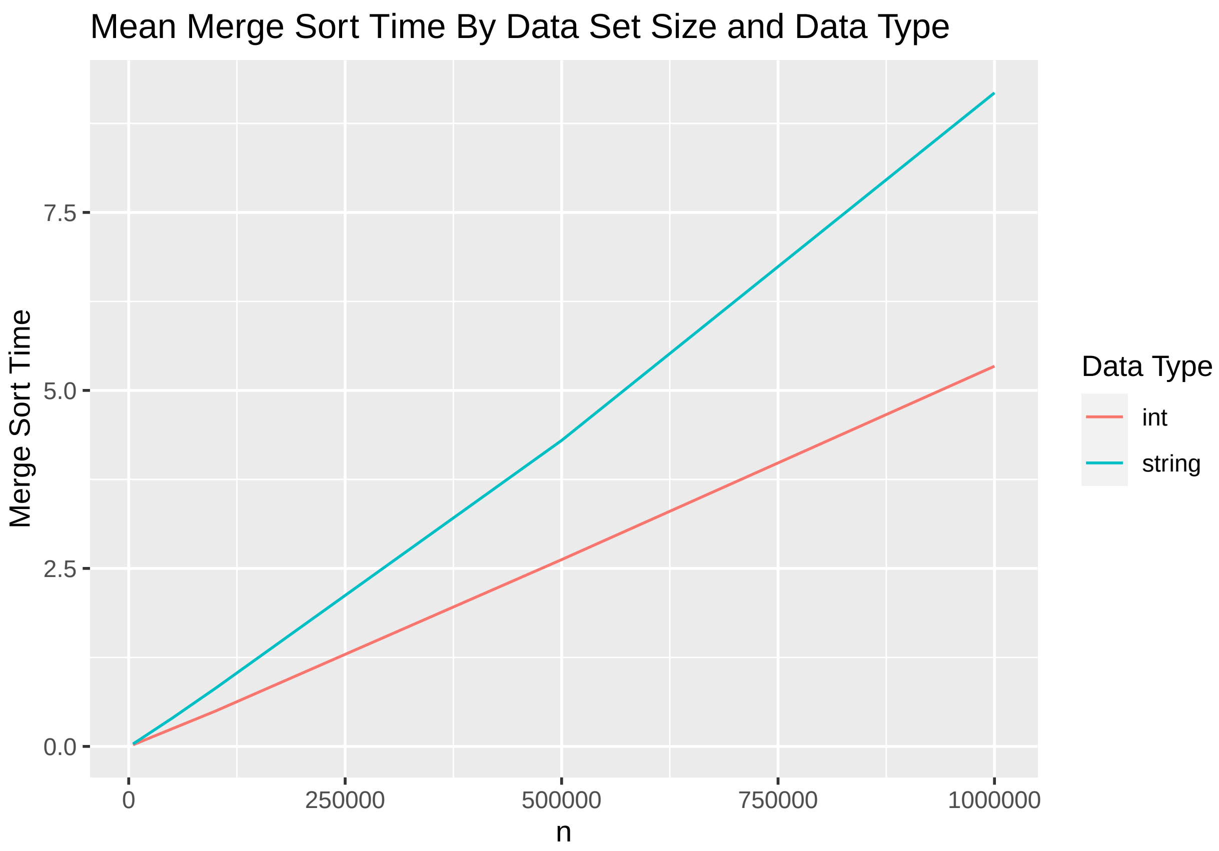


```
quickStrings = subset(quickTimes, var_type == "string")
ggplot(quickStrings, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Quick
  guides(color = guide_legend(title = "File Format"))
```



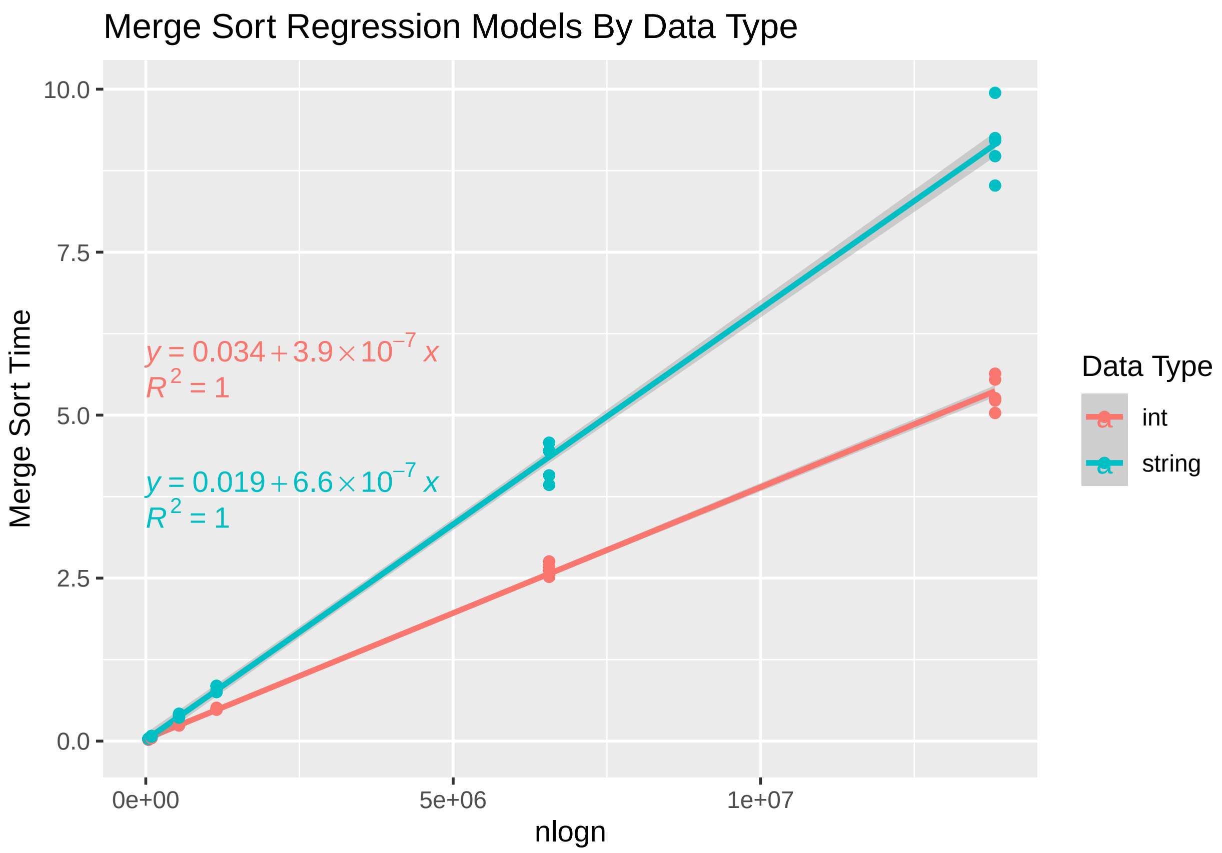
Merge Sort

```r
mergeTimes = aggregate(merge_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
mergeTimes2 = aggregate(merge_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(mergeTimes2, aes(x = size, y = merge_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Merge Sort Time By Data Set Size and Data Type", x = "n", y = "Merge Sort Time") +
  guides(color = guide_legend(title = "Data Type"))
```

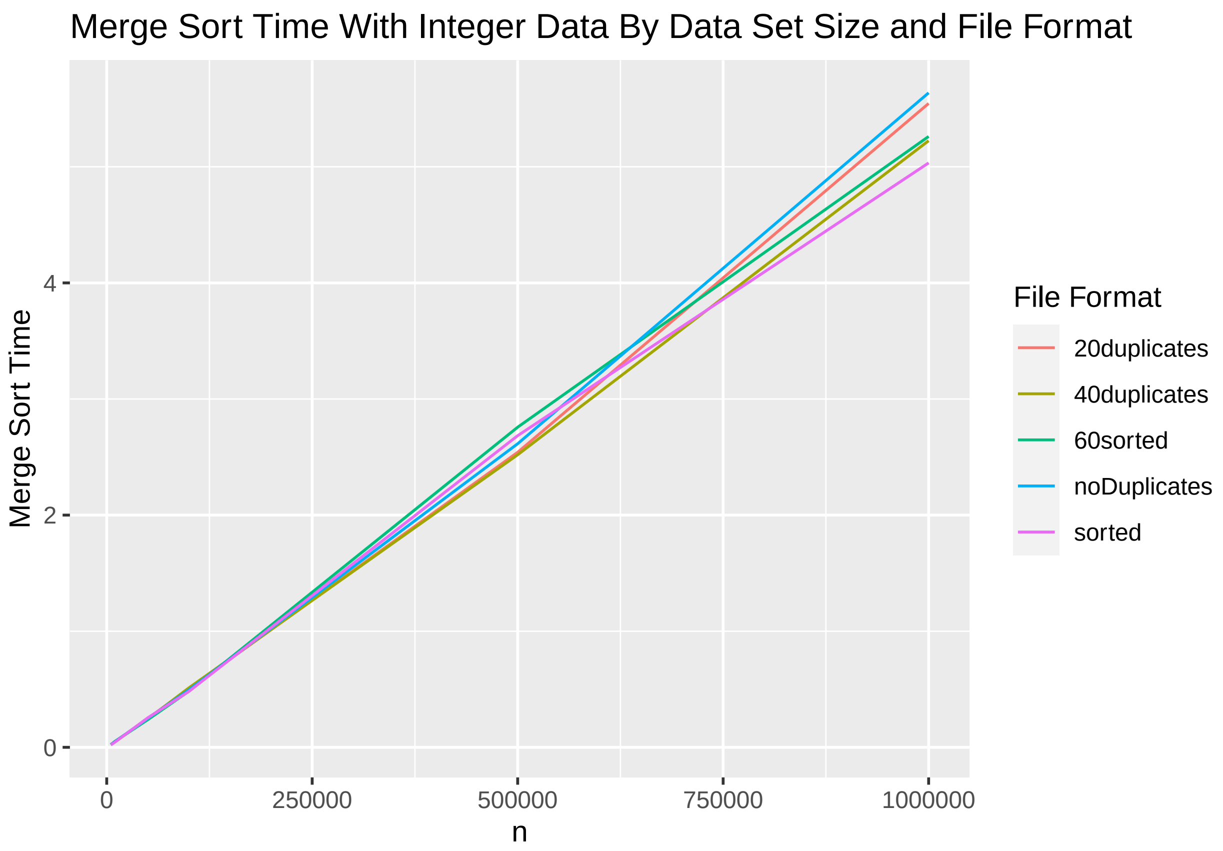


```
ggplot(mergeTimes, aes(x = nlogn, y = merge_time, color = var_type)) +
  labs(title = "Merge Sort Regression Models By Data Type", x = "nlogn", y = "Merge Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(6, 4)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(5.5, 3.5)) +
  guides(color = guide_legend(title = "Data Type"))
```

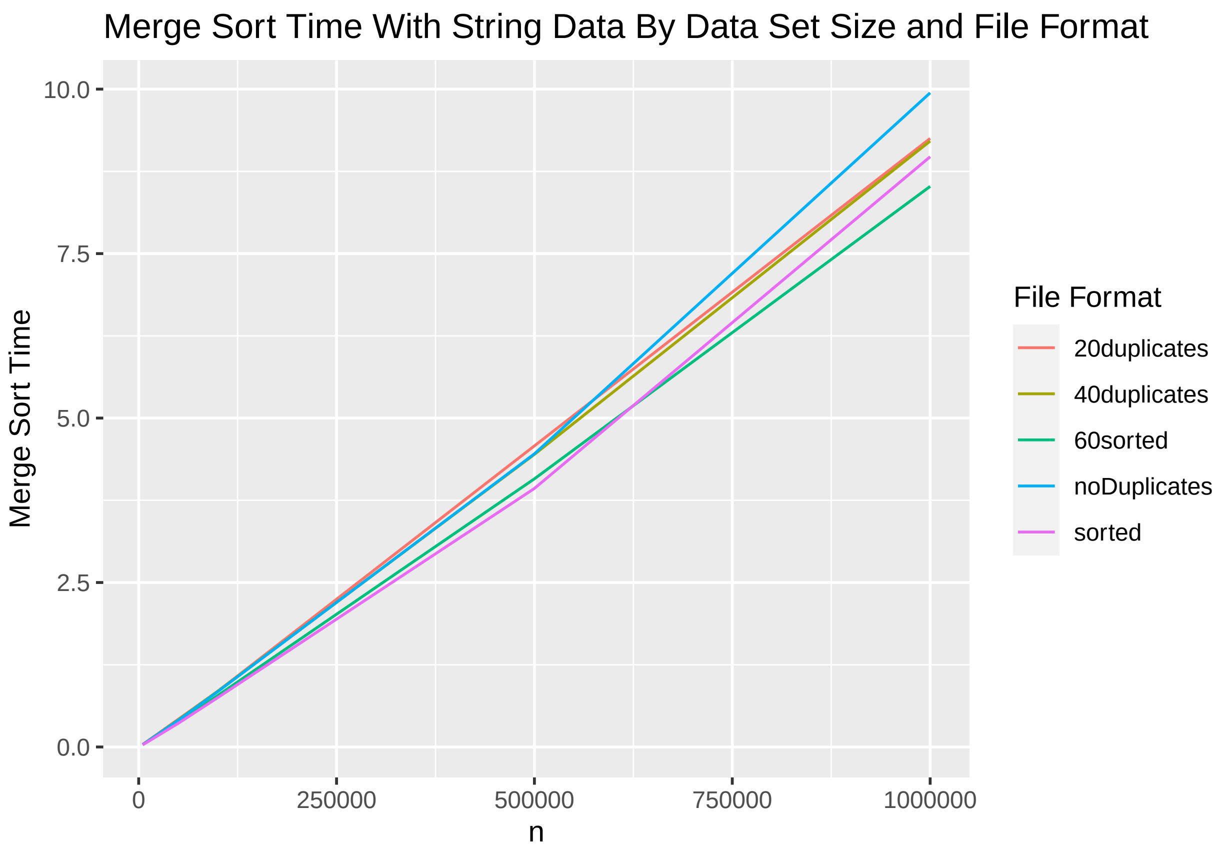
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
mergeInts = subset(mergeTimes, var_type == "int")
ggplot(mergeInts, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Merge
  guides(color = guide_legend(title = "File Format"))
```

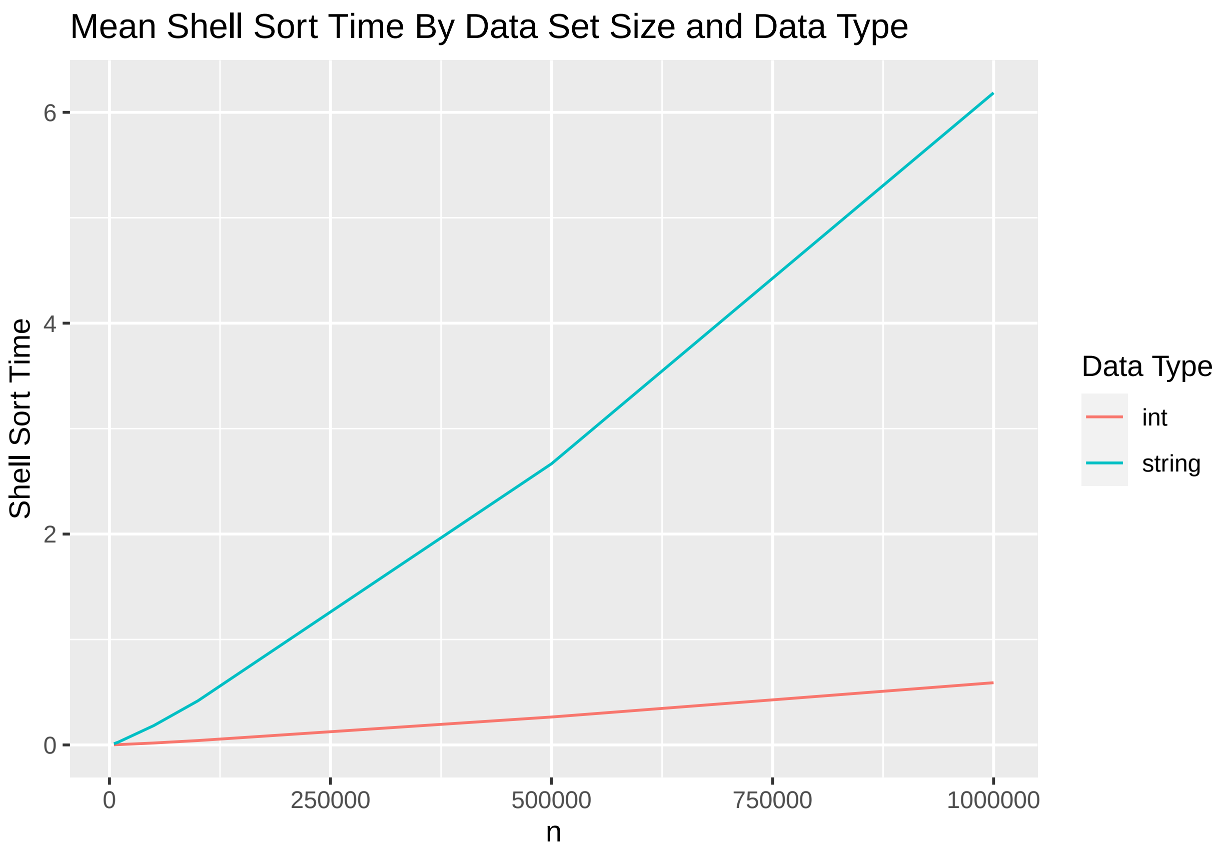


```
mergeStrings = subset(mergeTimes, var_type == "string")
ggplot(mergeStrings, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Merge
  guides(color = guide_legend(title = "File Format"))
```



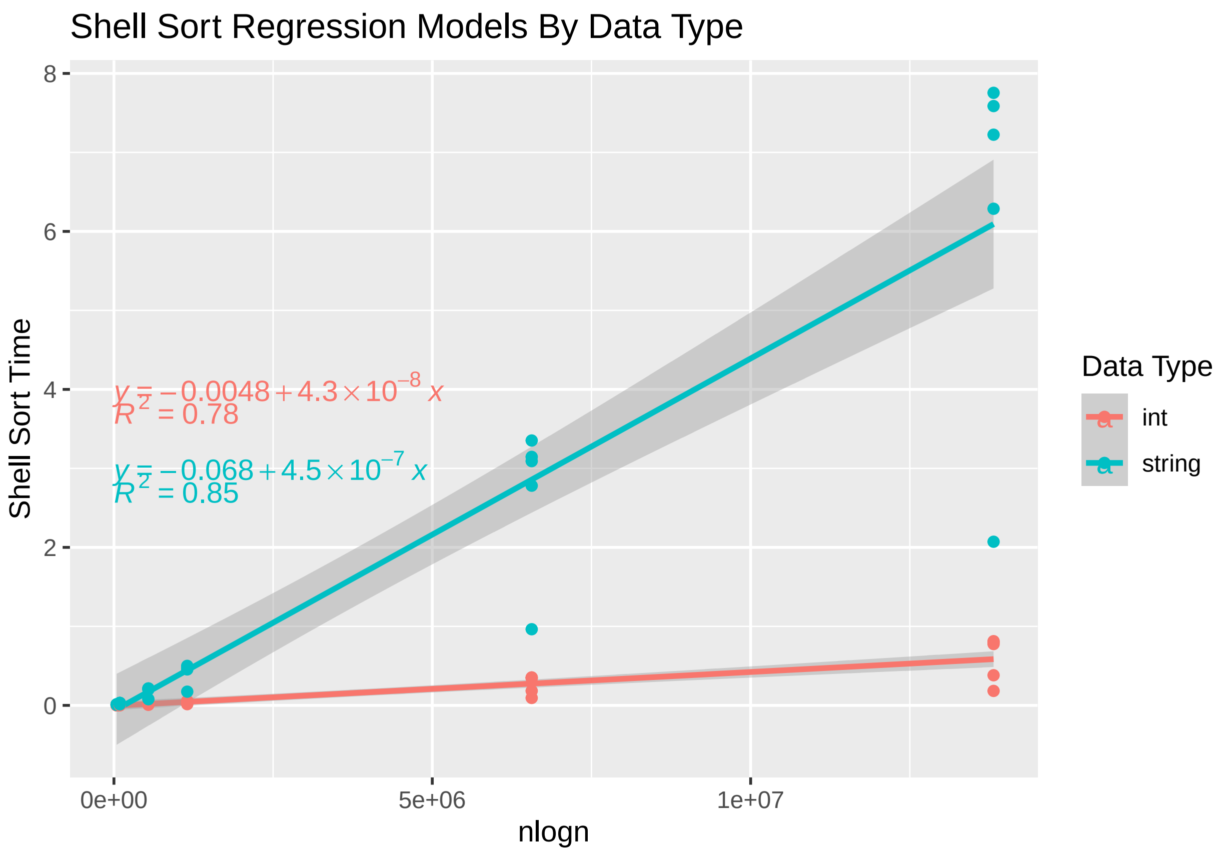
Shell Sort

```
shellTimes = aggregate(shell_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
shellTimes2 = aggregate(shell_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(shellTimes2, aes(x = size, y = shell_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Shell Sort Time By Data Set Size and Data Type", x = "n", y = "Shell Sort Time") +
  guides(color = guide_legend(title = "Data Type"))
```

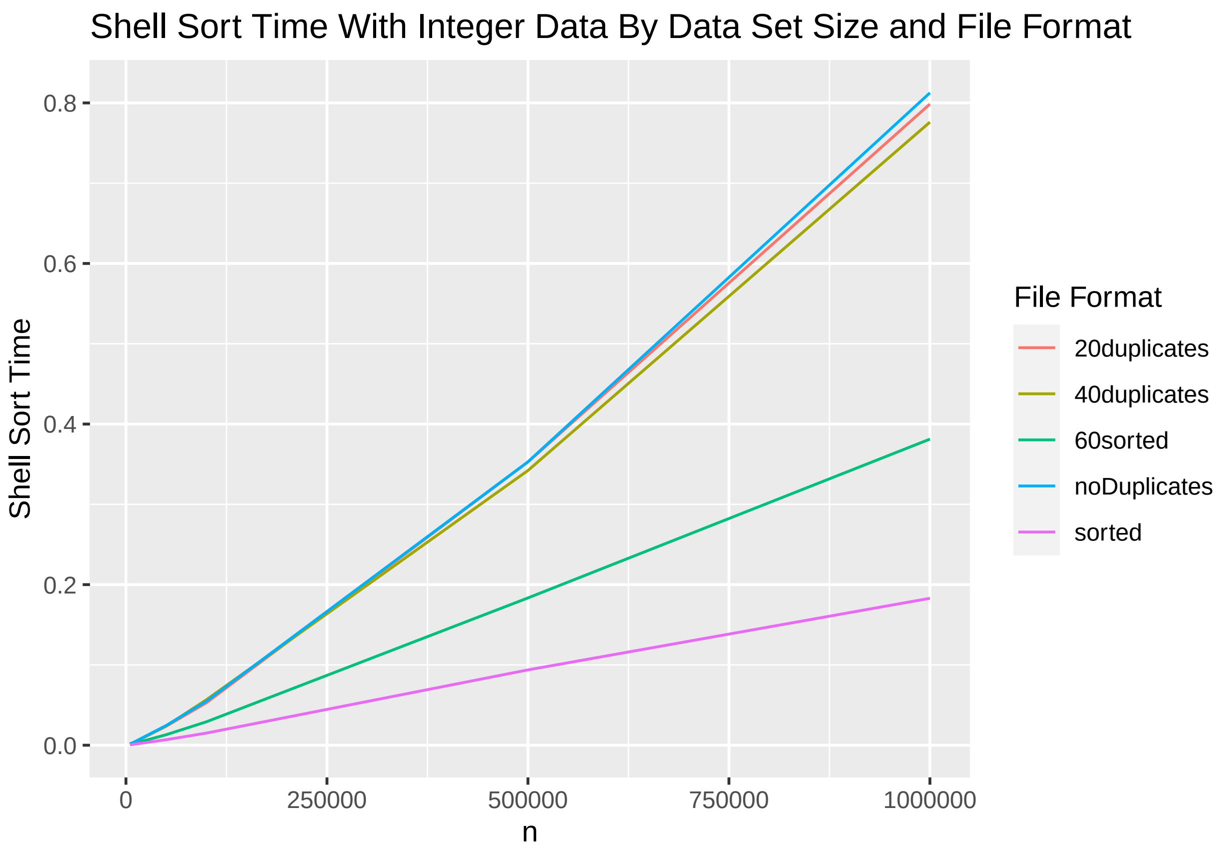


```
ggplot(shellTimes, aes(x = nlogn, y = shell_time, color = var_type)) +
  labs(title = "Shell Sort Regression Models By Data Type", x = "nlogn", y = "Shell Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(3.75, 2.75)) +
  guides(color = guide_legend(title = "Data Type"))
```

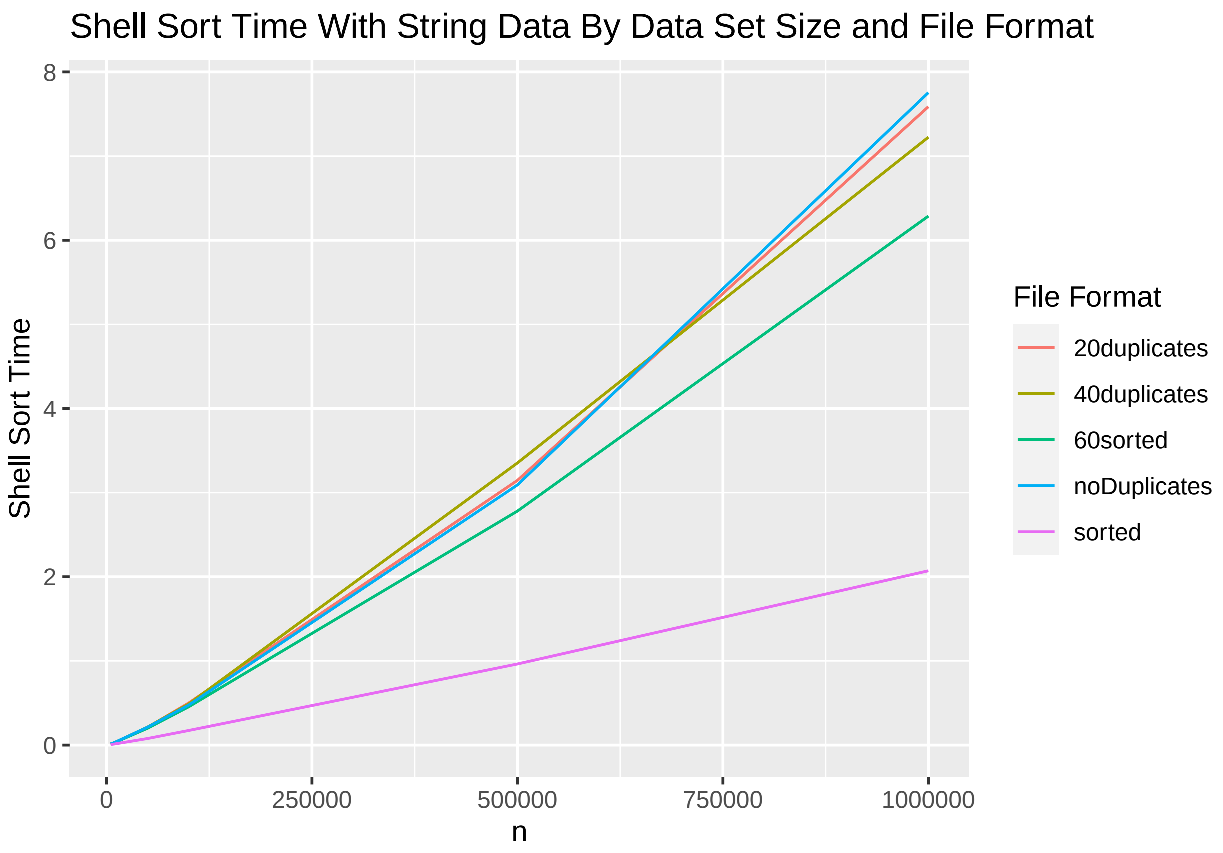
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
shellInts = subset(shellTimes, var_type == "int")
ggplot(shellInts, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Shel
  guides(color = guide_legend(title = "File Format"))
```

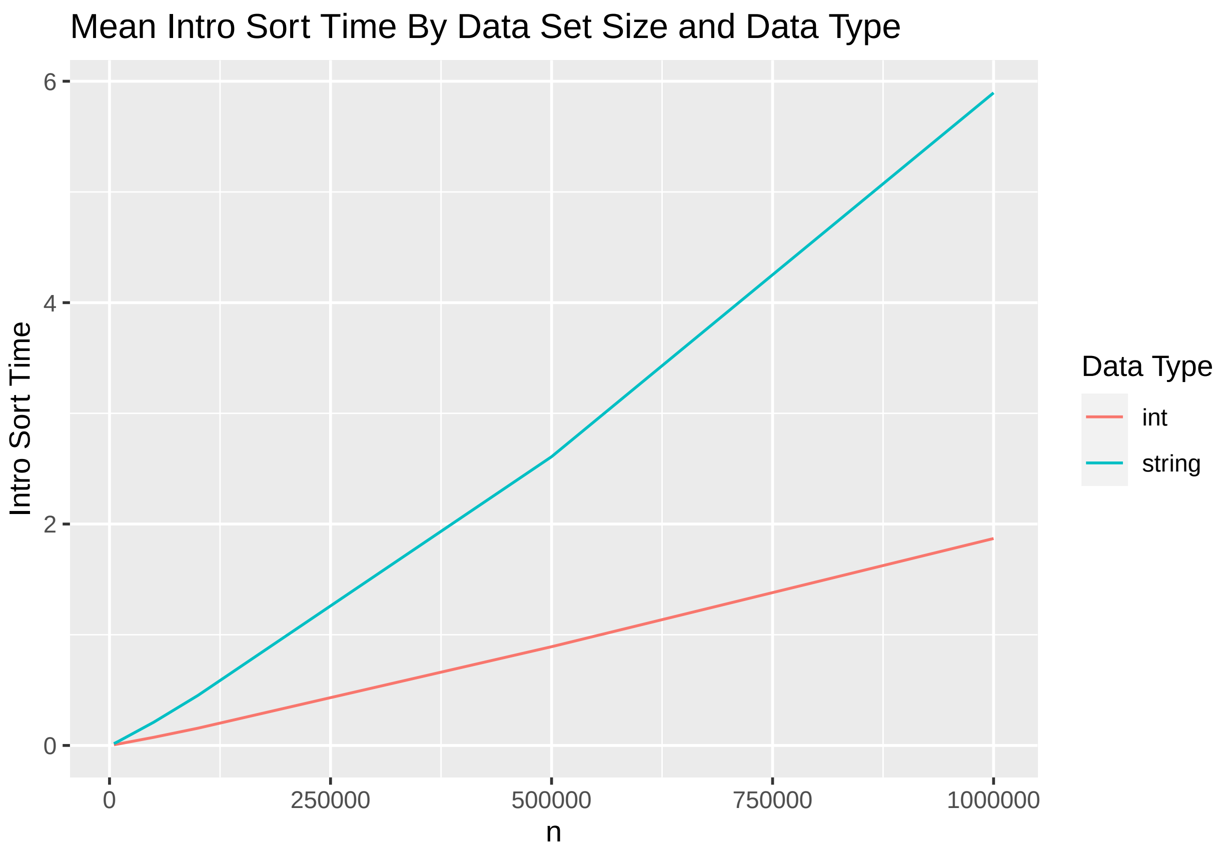


```
shellStrings = subset(shellTimes, var_type == "string")
ggplot(shellStrings, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Shell
  guides(color = guide_legend(title = "File Format"))
```



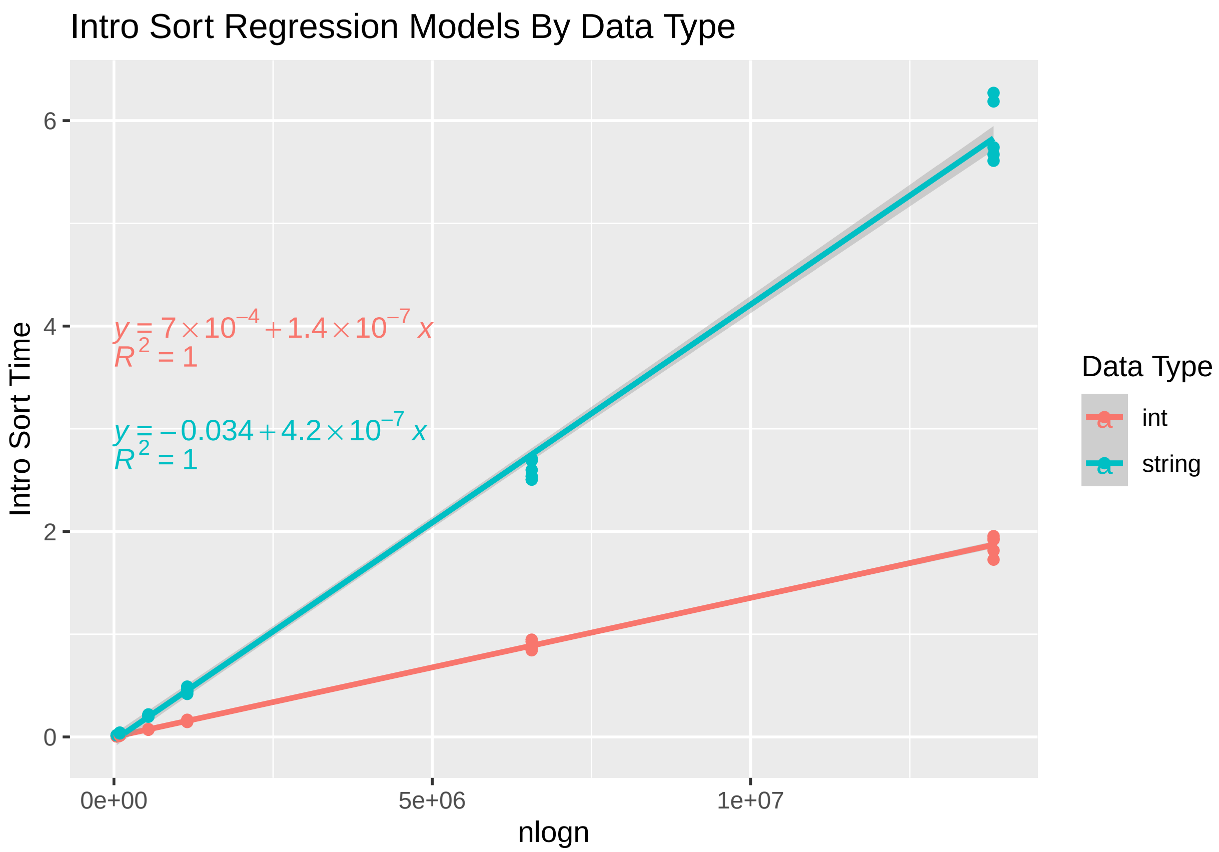
Intro Sort

```
introTimes = aggregate(intro_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
introTimes2 = aggregate(intro_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(introTimes2, aes(x = size, y = intro_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Intro Sort Time By Data Set Size and Data Type", x = "n", y = "Intro Sort Time") +
  guides(color = guide_legend(title = "Data Type"))
```

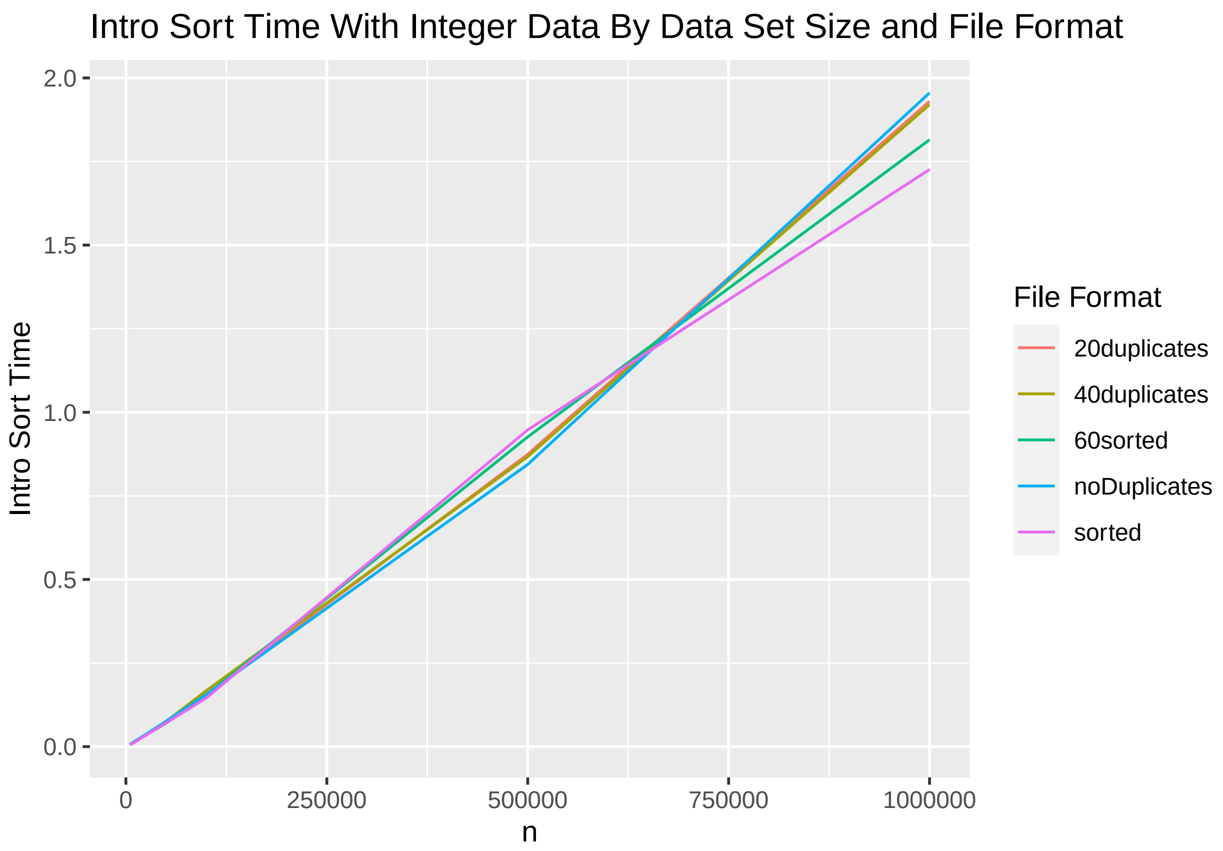


```
ggplot(introTimes, aes(x = nlogn, y = intro_time, color = var_type)) +
  labs(title = "Intro Sort Regression Models By Data Type", x = "nlogn", y = "Intro Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(3.75, 2.75)) +
  guides(color = guide_legend(title = "Data Type"))
```

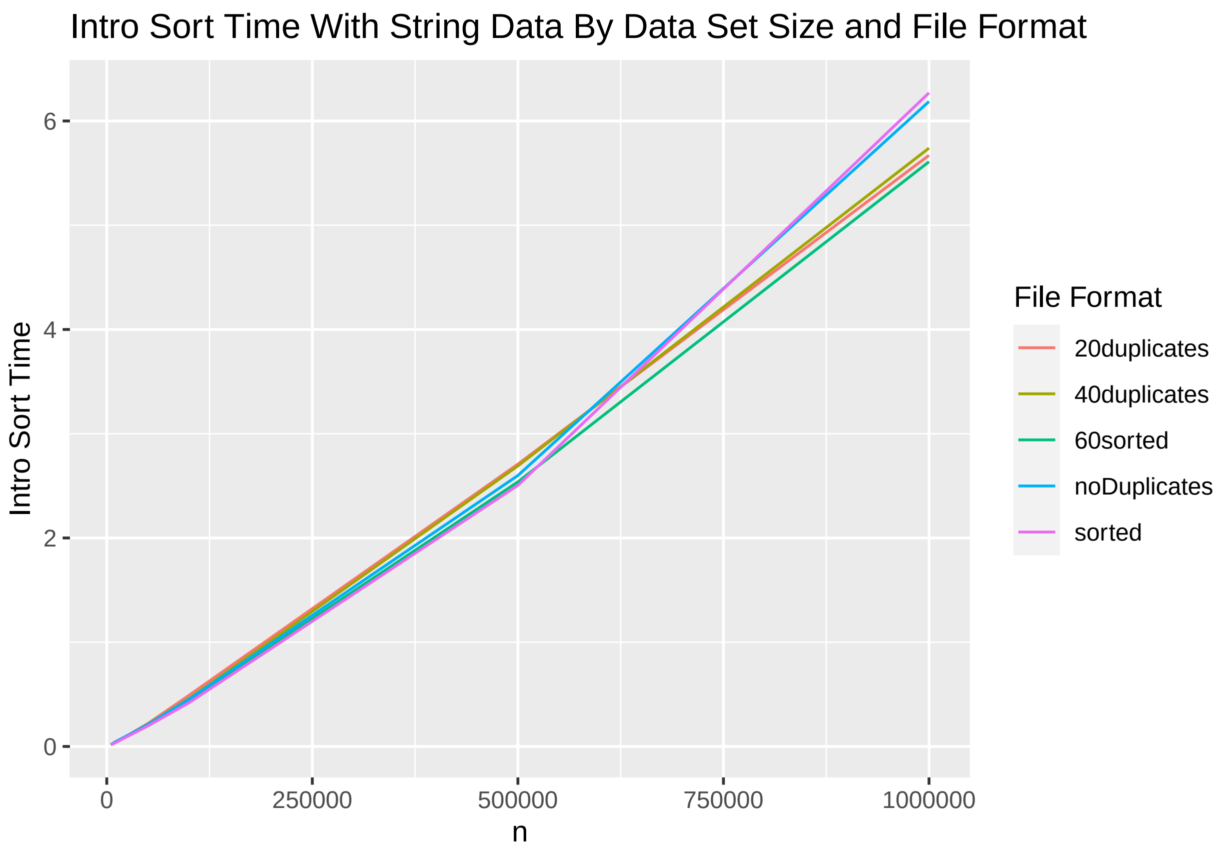
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```r
introInts = subset(introTimes, var_type == "int")
ggplot(introInts, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))
```

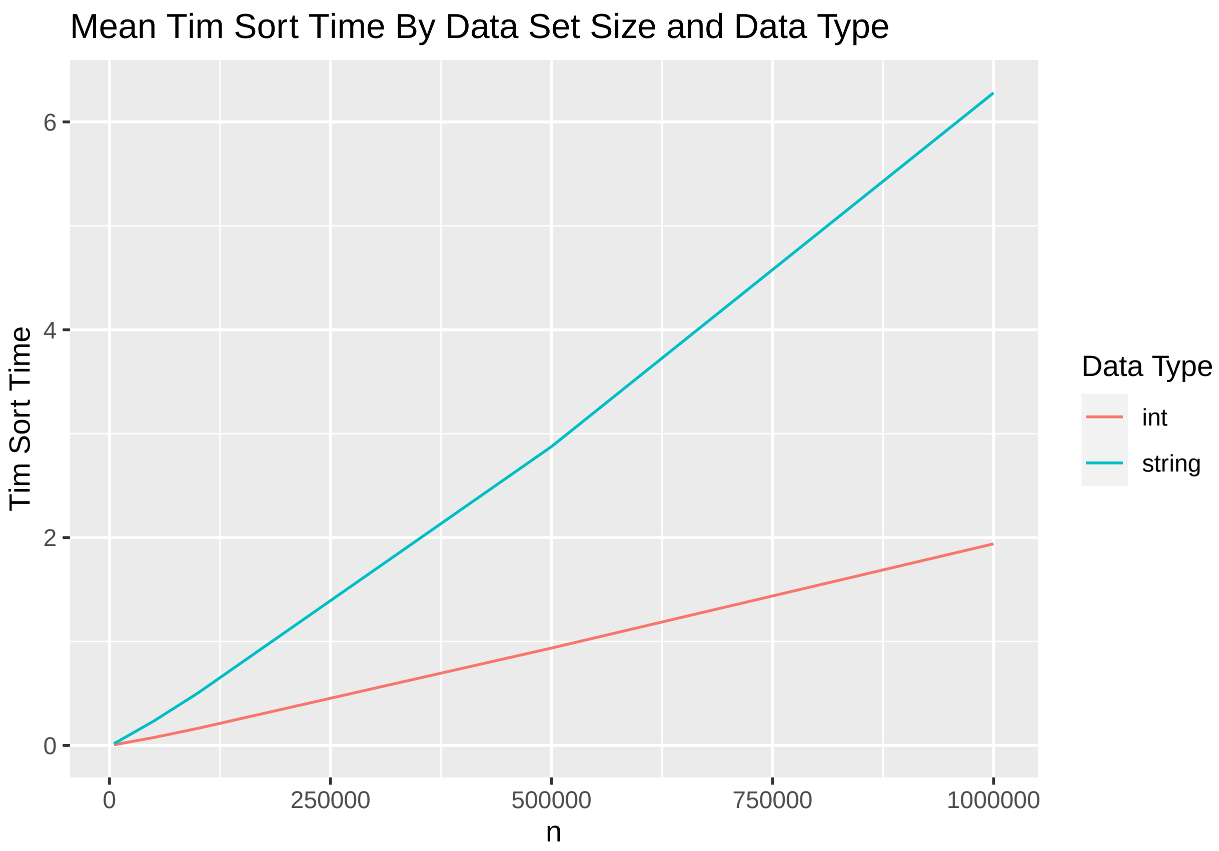


```
introStrings = subset(introTimes, var_type == "string")
ggplot(introStrings, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))
```



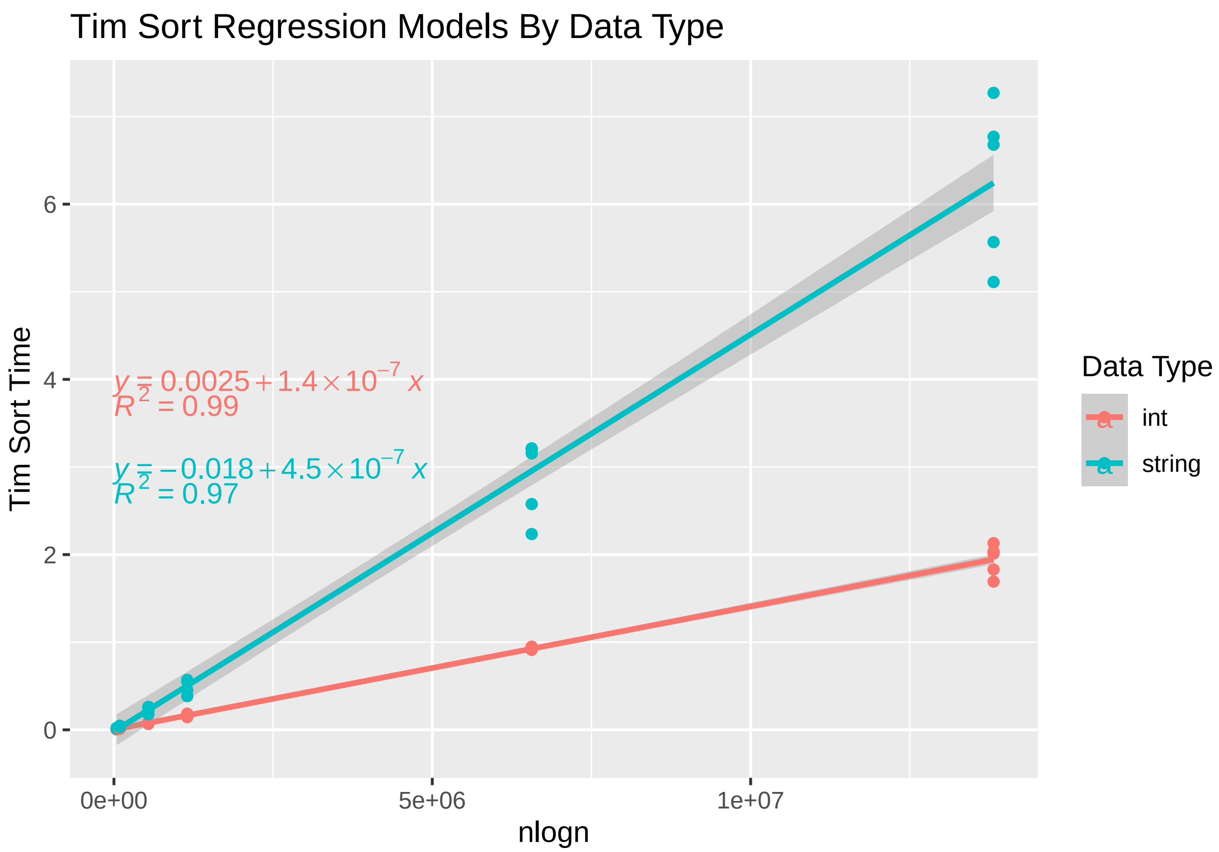
Tim Sort

```
timTimes = aggregate(tim_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
timTimes2 = aggregate(tim_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(timTimes2, aes(x = size, y = tim_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Tim Sort Time By Data Set Size and Data Type", x = "n", y = "Tim Sort Time") +
  guides(color = guide_legend(title = "Data Type"))
```

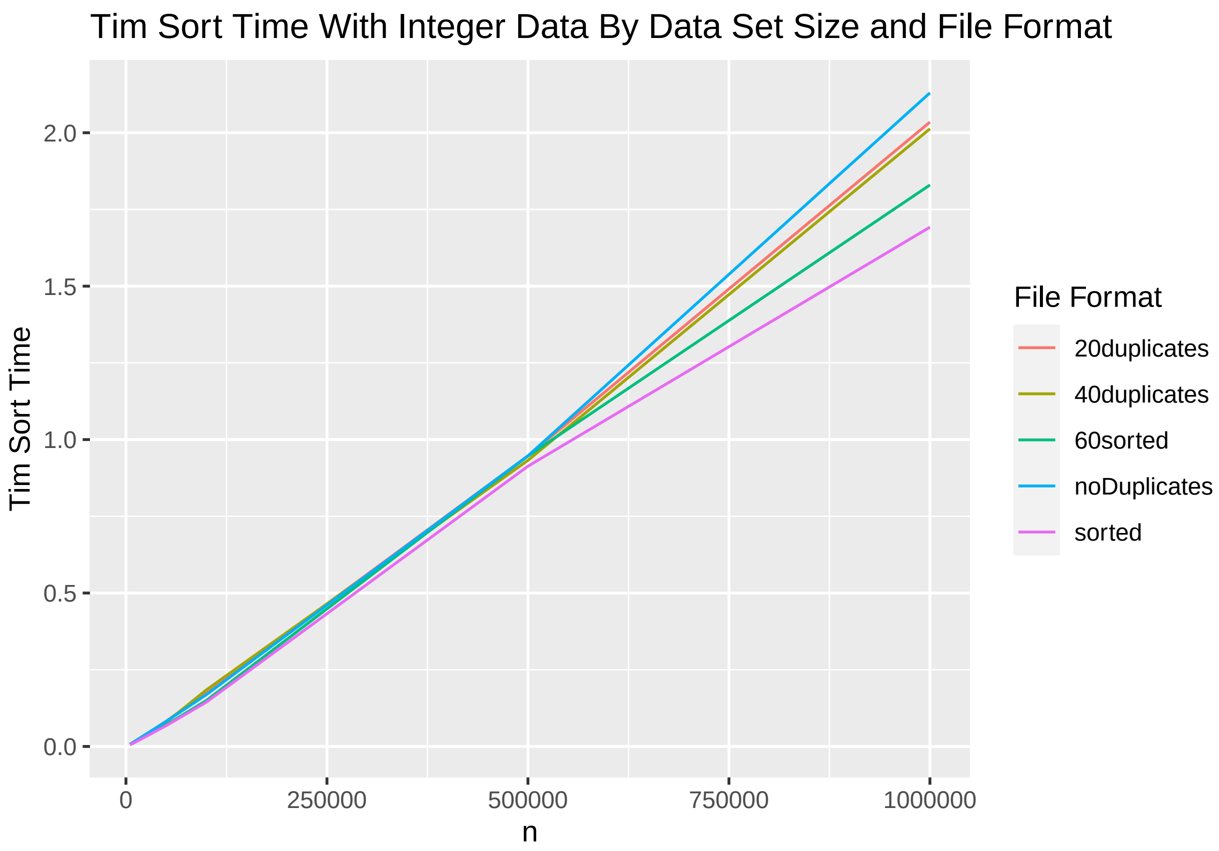


```
ggplot(timTimes, aes(x = nlogn, y = tim_time, color = var_type)) +
  labs(title = "Tim Sort Regression Models By Data Type", x = "nlogn", y = "Tim Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..), label.x=0, label.y=c(3.75, 2.75)) +
  guides(color = guide_legend(title = "Data Type"))
```

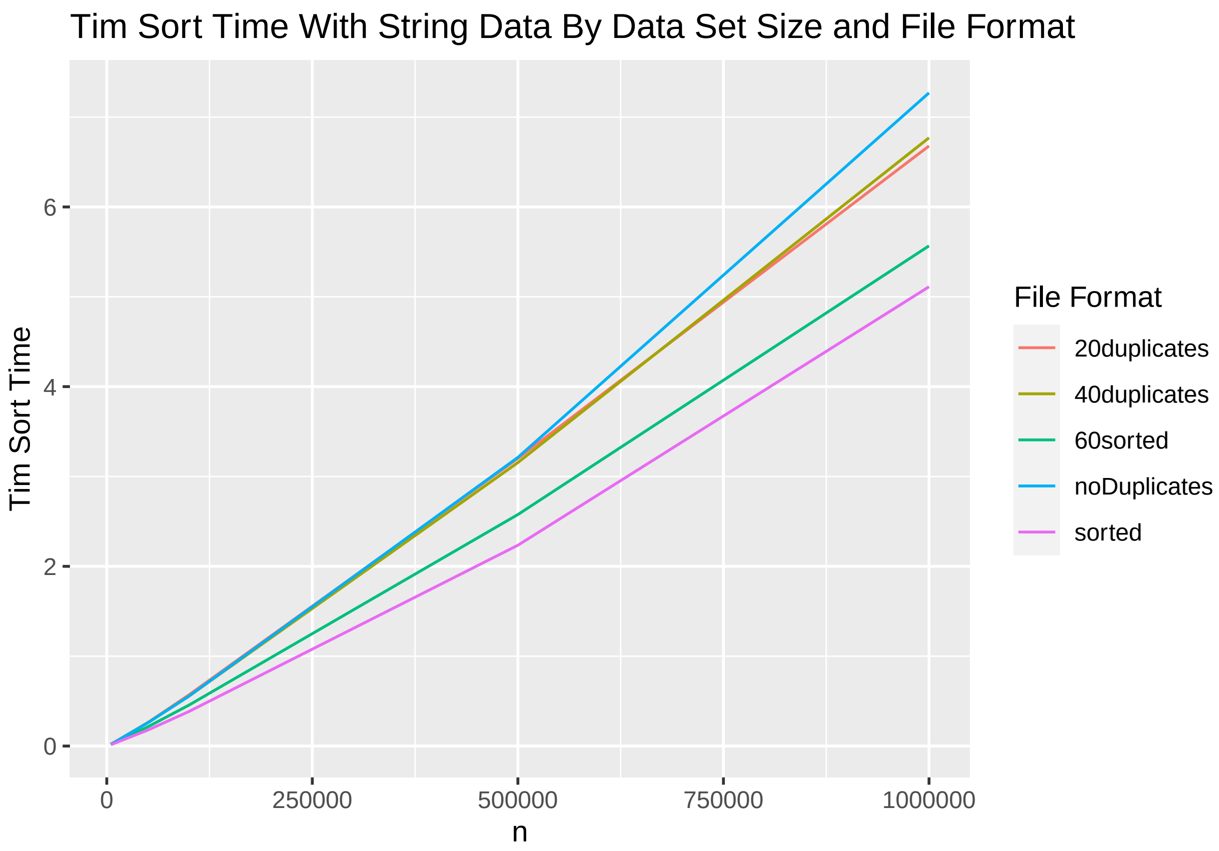
```
## 'geom_smooth()' using formula 'y ~ x'
```



```r
timInts = subset(timTimes, var_type == "int")
ggplot(timInts, aes(x = size, y = tim_time, color = format)) +
  geom_line() +
  labs(title = "Tim Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Tim So
  guides(color = guide_legend(title = "File Format"))
```



```
timStrings = subset(timTimes, var_type == "string")
ggplot(timStrings, aes(x = size, y = tim_time, color = format)) +
  geom_line() +
  labs(title = "Tim Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Tim Sort
  guides(color = guide_legend(title = "File Format"))
```

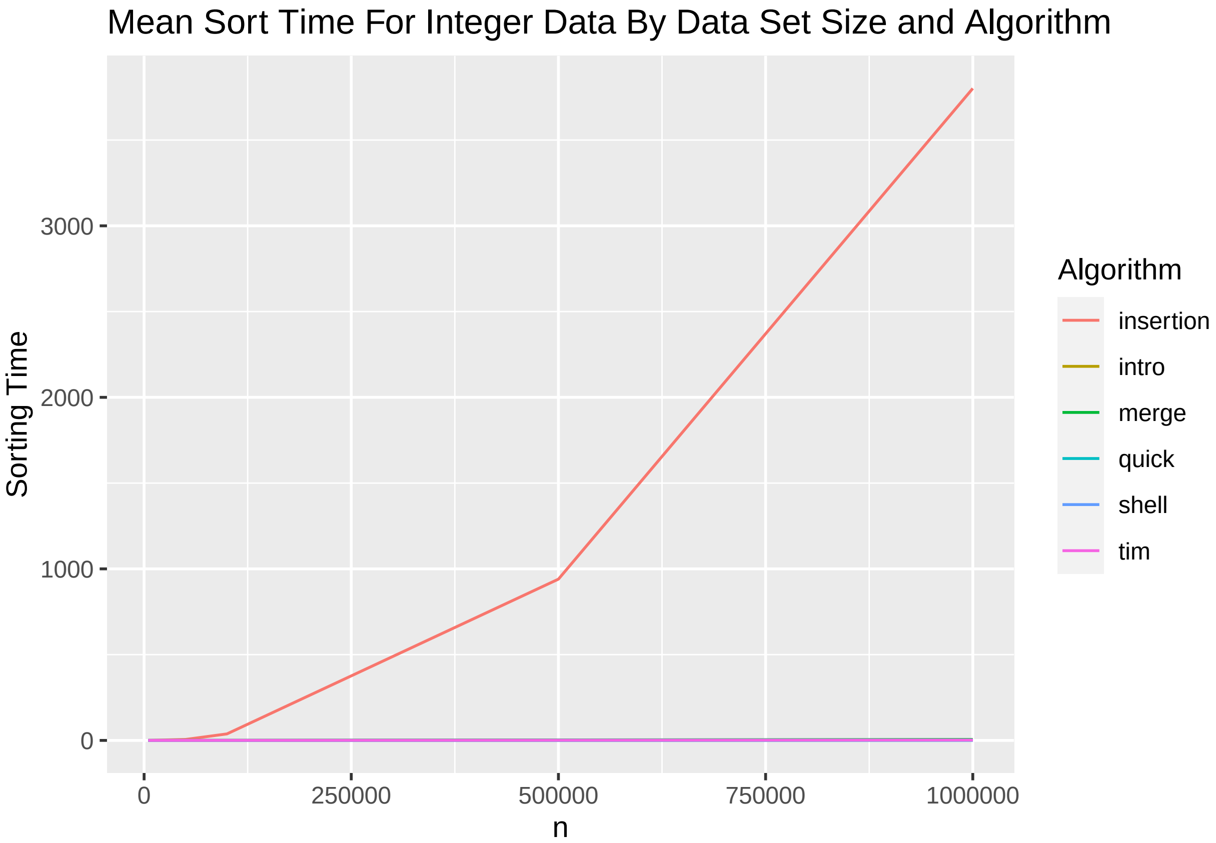


Algorithm Comparison

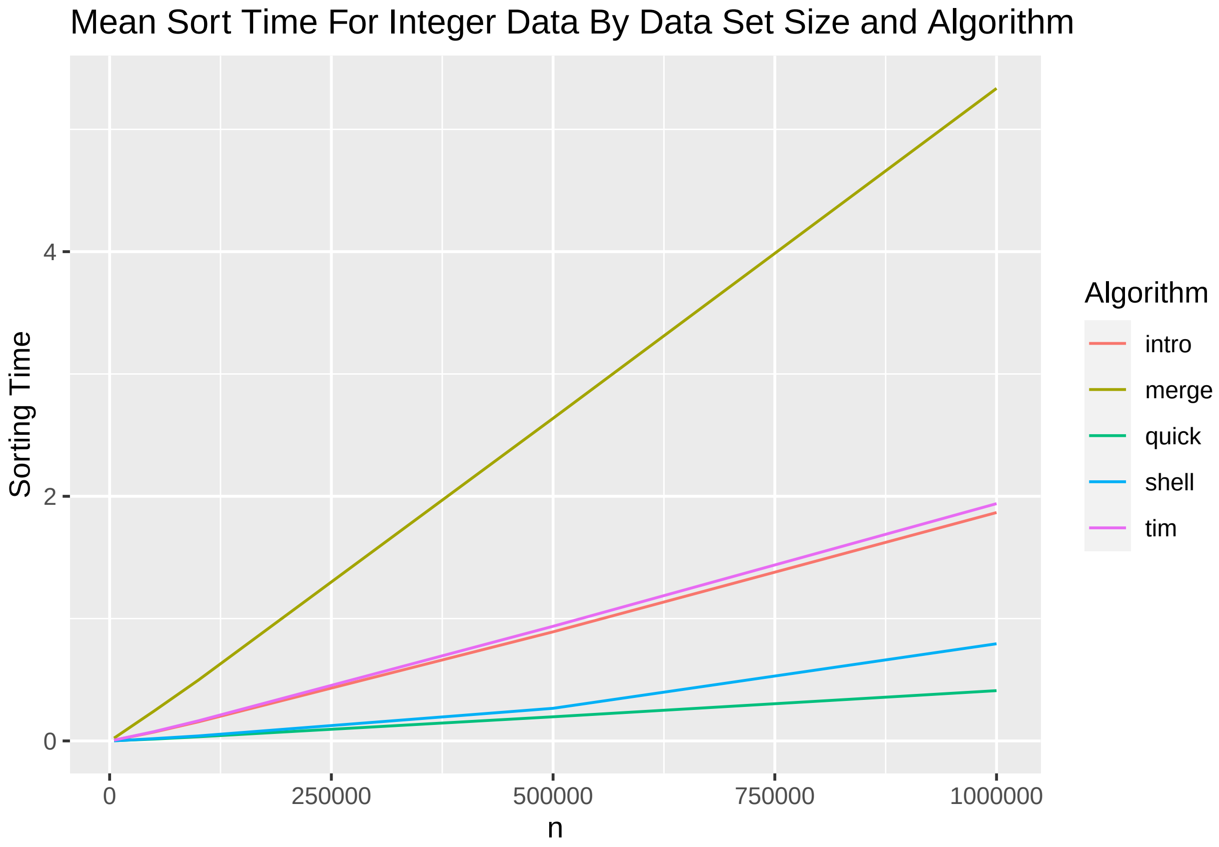
```
data2 = matrix(ncol = 5, nrow = 360)
for (i in 1:6) {
  for (j in 1:60) {
    data2[i * j, 1] = data[j, 1]
    data2[i * j, 2] = data[j, 2]
    data2[i * j, 3] = data[j, 3]
    data2[i * j, 4] = data[j, 3 + i]
    if (i == 1) {
      data2[i * j, 5] = "insertion"
    } else if (i == 2) {
      data2[i * j, 5] = "quick"
    } else if (i == 3) {
      data2[i * j, 5] = "merge"
    } else if (i == 4) {
      data2[i * j, 5] = "shell"
    } else if (i == 5) {
      data2[i * j, 5] = "intro"
    } else {
      data2[i * j, 5] = "tim"
    }
  }
}
```

```
data2 = data.frame(data2)
colnames(data2) = c("var_type", "size", "format", "time", "algorithm")
data2 = transform(data2, time = as.numeric(time))
data2 = transform(data2, size = as.numeric(size))
```

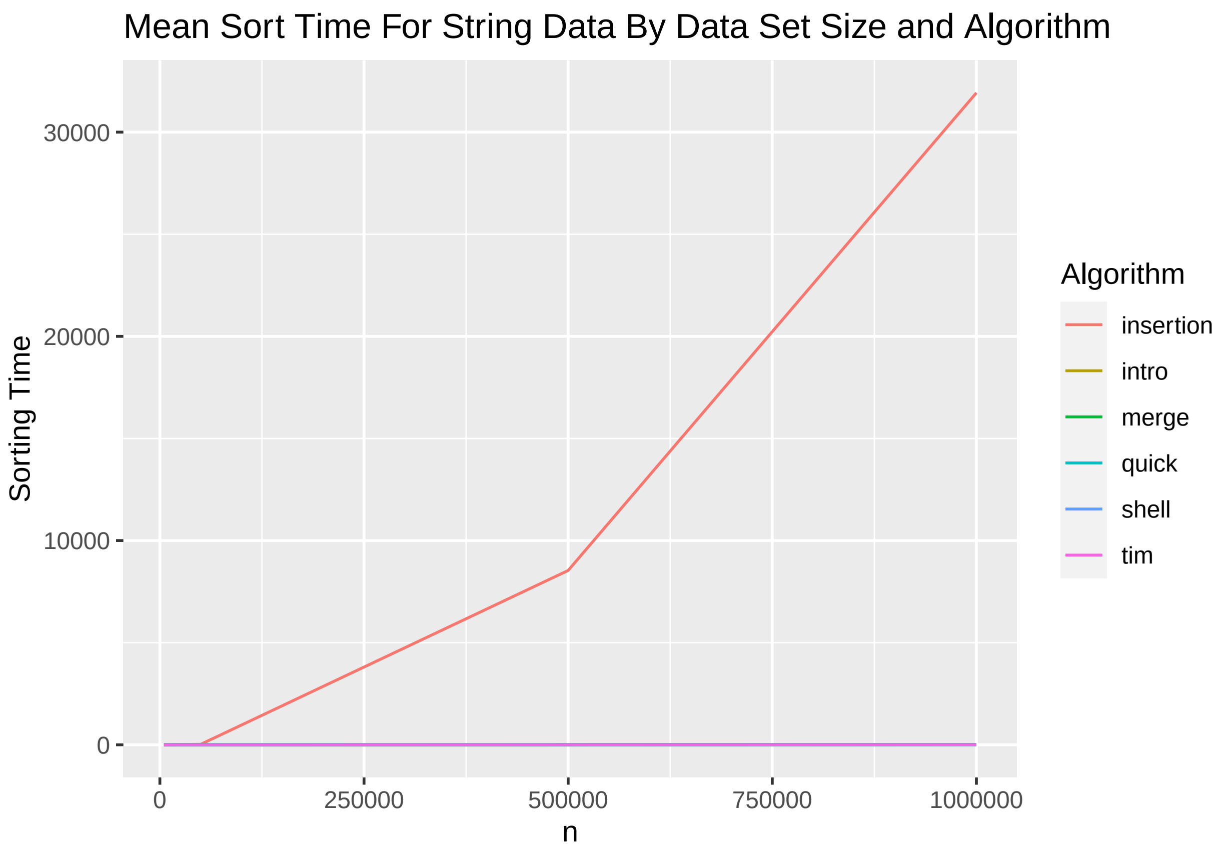
```
integerData = subset(data2, var_type == "int")
integerTimes = aggregate(time ~ algorithm + size, data = integerData, FUN = mean)
ggplot(integerTimes, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x = "n", y = "Sorting T
  guides(color = guide_legend(title = "Algorithm"))
```



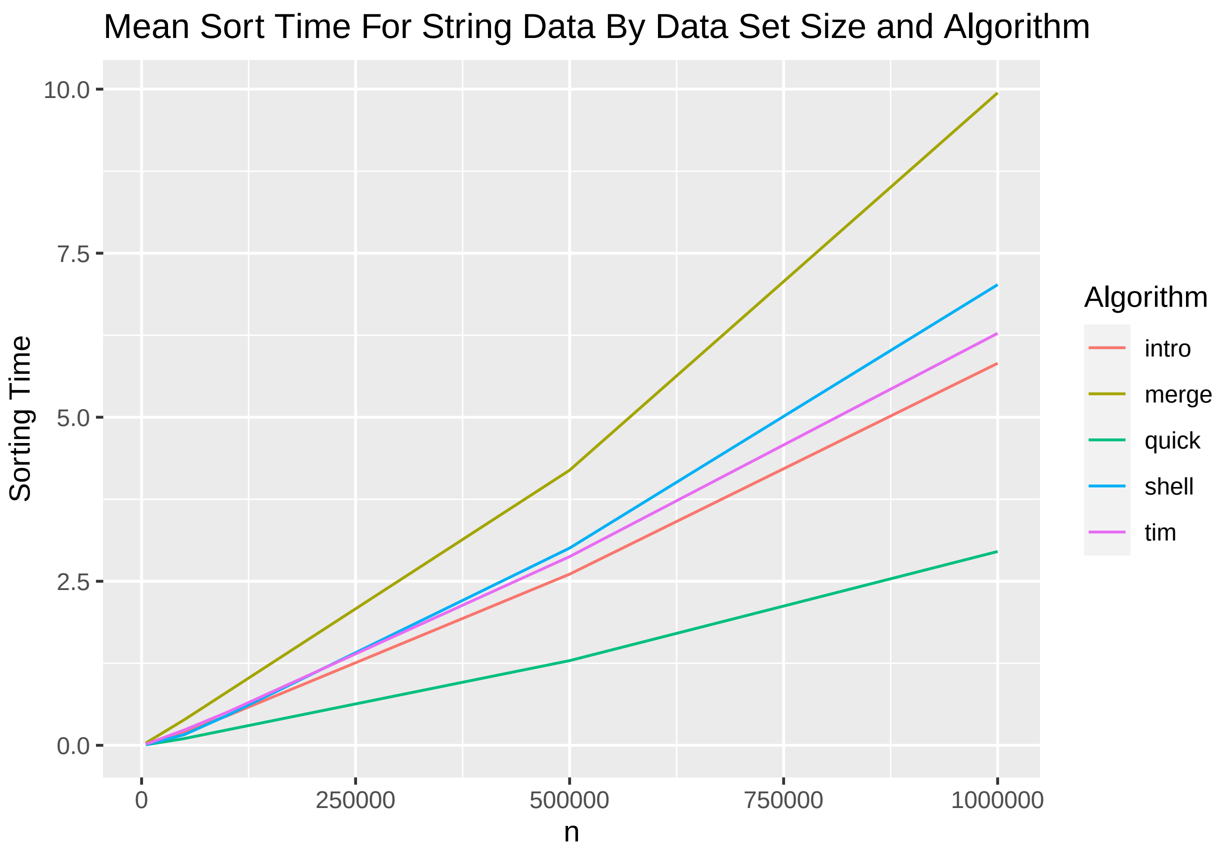
```
integerTimes2 = subset(integerTimes, algorithm != "insertion")
ggplot(integerTimes2, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x = "n", y = "Sorting T
  guides(color = guide_legend(title = "Algorithm"))
```



```
stringData = subset(data2, var_type == "string")
stringTimes = aggregate(time ~ algorithm + size, data = stringData, FUN = mean)
ggplot(stringTimes, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For String Data By Data Set Size and Algorithm", x = "n", y = "Sorting Ti
  guides(color = guide_legend(title = "Algorithm"))
```



```
stringTimes2 = subset(stringTimes, algorithm != "insertion")
ggplot(stringTimes2, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For String Data By Data Set Size and Algorithm", x = "n", y = "Sorting Ti
  guides(color = guide_legend(title = "Algorithm"))
```


Mean Sort Time For String Data By Data Set Size and Algorithm
Sorting Time
10.0
7.5
5.0
2.5
0.0
0
250000
500000
750000
1000000
n
Algorithm
intro
merge
quick
shell
tim