Program 2 Graph Analysis

Ryan Schaefer and Wes Anderson

Create Dataset

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

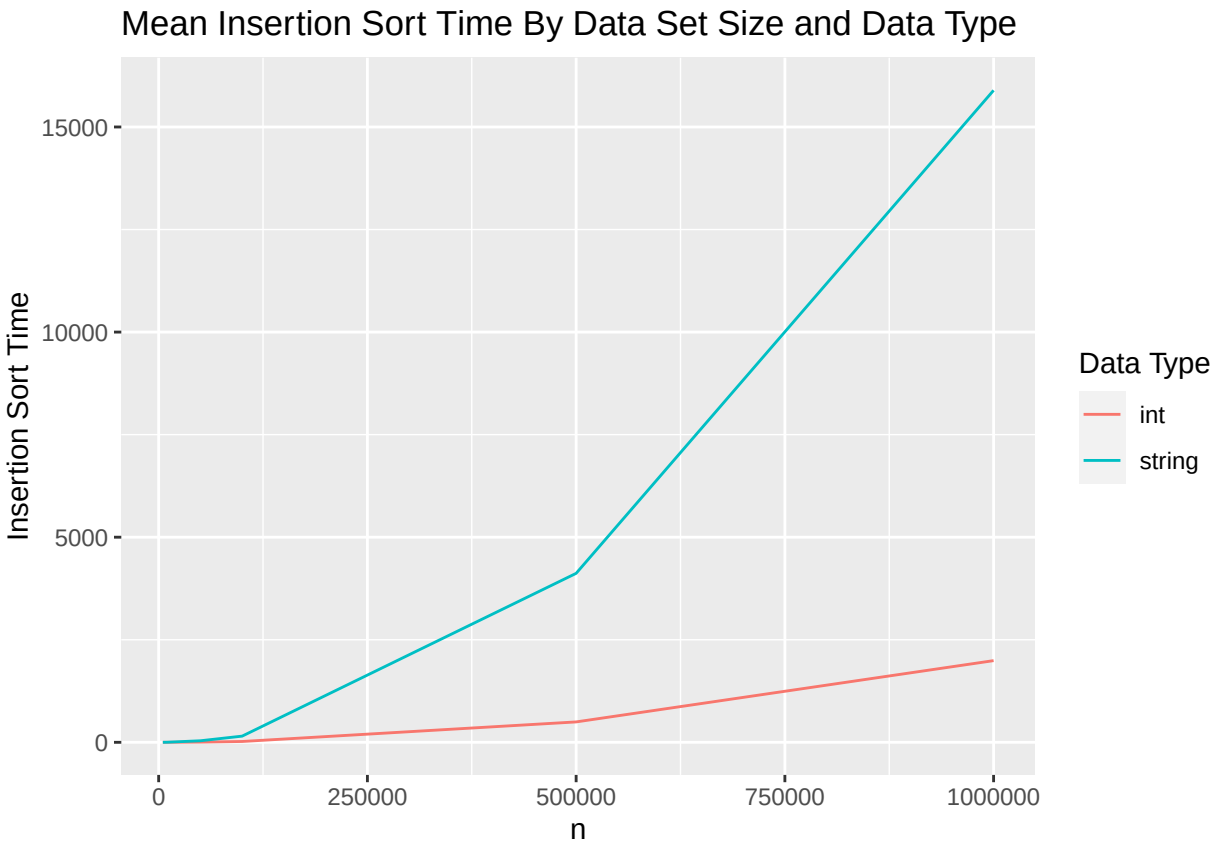
```
##    var_type    size       format insertion_time  quick_time merge_time
## 1       int  500000 noDuplicates    7.95005e+02 0.164816000 2.02272667
## 2       int 1000000 40duplicates    3.16407e+03 0.313102667 4.07061667
## 3       int  100000 40duplicates    3.15688e+01 0.028900267 0.40771467
## 4       int   10000 40duplicates    3.32033e-01 0.002583923 0.03599610
## 5       int   50000       sorted    6.90530e-04 0.010440873 0.19003033
## 6       int   50000 20duplicates    7.85778e+00 0.015082900 0.19768167
## 7       int    5000 noDuplicates    8.12401e-02 0.001328160 0.01882400
## 8       int  500000       sorted    7.13136e-03 0.106529433 2.00223667
## 9       int  500000     60sorted    1.26520e+02 0.130218333 2.02036667
## 10      int   10000     60sorted    4.75302e-02 0.002104807 0.03639520
## 11      int 1000000 noDuplicates    3.13320e+03 0.338367333 4.32009000
## 12      int 1000000 20duplicates    3.16243e+03 0.333363000 4.27491000
## 13      int   50000 noDuplicates    7.92707e+00 0.014422000 0.20717133
## 14      int    5000     60sorted    1.33967e-02 0.001247927 0.02053703
## 15      int    5000       sorted    6.16000e-05 0.001098407 0.01817163
## 16      int  100000 20duplicates    3.16373e+01 0.030057100 0.40908700
## 17      int   50000     60sorted    1.26396e+00 0.011773033 0.19947167
## 18      int   10000 noDuplicates    3.23680e-01 0.002838517 0.03884203
## 19      int  500000 20duplicates    7.86741e+02 0.161247667 2.08697333
## 20      int  500000 40duplicates    7.76092e+02 0.158967000 2.09886000
## 21      int 1000000       sorted    1.22738e-02 0.214287333 4.07474000
## 22      int    5000 20duplicates    7.45130e-02 0.001358083 0.01915807
## 23      int  100000 noDuplicates    3.10339e+01 0.029615267 0.40823933
## 24      int   50000 40duplicates    7.75182e+00 0.016053433 0.20144567
## 25      int   10000 20duplicates    3.18130e-01 0.002806347 0.03943440
## 26      int  100000       sorted    1.48089e-03 0.020476367 0.39963700
## 27      int  100000     60sorted    5.00708e+00 0.024081933 0.40652467
## 28      int   10000       sorted    1.48140e-04 0.001990017 0.03820463
## 29      int    5000 40duplicates    8.49797e-02 0.001362817 0.01942190
## 30      int 1000000     60sorted    4.96103e+02 0.269172000 4.19366333
## 31   string   50000       sorted    5.34647e-03 0.080588467 0.26832967
## 32   string  500000 20duplicates    5.14512e+03 0.997793000 3.34057333
## 33   string   50000 20duplicates    5.10532e+01 0.080015633 0.30973933
```

```
## 34   string   10000 40duplicates    2.04046e+00 0.013992533 0.05887447
## 35   string   10000     60sorted    1.29043e+00 0.014251200 0.05484163
## 36   string  100000       sorted    1.10993e-02 0.165743333 0.57058000
## 37   string    5000 40duplicates    5.10891e-01 0.006785933 0.02772243
## 38   string  500000     60sorted    3.28954e+03 0.977737333 3.08546333
## 39   string   50000 noDuplicates    5.16270e+01 0.082316533 0.31142533
## 40   string  500000 40duplicates    5.14260e+03 1.060223333 3.35945333
## 41   string    5000 20duplicates    5.16617e-01 0.006900917 0.02842427
## 42   string  100000 noDuplicates    2.05698e+02 0.174706333 0.63412933
## 43   string    5000 noDuplicates    5.21201e-01 0.006469280 0.02738830
## 44   string  100000     60sorted    1.31478e+02 0.177512667 0.59791533
## 45   string 1000000 20duplicates    2.09586e+04 2.196250000 6.96096667
## 46   string   10000 noDuplicates    2.05775e+00 0.018324200 0.05840567
## 47   string 1000000 noDuplicates    2.34202e+04 2.673316667 8.51005667
## 48   string 1000000       sorted    1.46393e-01 2.241120000 8.10947333
## 49   string  500000 noDuplicates    7.01488e+03 1.022823333 3.41494000
## 50   string  100000 40duplicates    2.09063e+02 0.188036667 0.64023133
## 51   string    5000     60sorted    3.25228e-01 0.006375277 0.02737953
## 52   string 1000000     60sorted    1.41741e+04 2.176563333 6.47977000
## 53   string    5000       sorted    4.50960e-04 0.005782653 0.02583847
## 54   string  100000 20duplicates    2.05982e+02 0.174944333 0.64222567
## 55   string   10000 20duplicates    2.03632e+00 0.019363933 0.06660247
## 56   string   10000       sorted    9.79630e-04 0.013212300 0.05443057
## 57   string  500000       sorted    4.75168e-02 0.982234000 2.96417000
## 58   string   50000 40duplicates    5.13650e+01 0.083264233 0.32026000
## 59   string   50000     60sorted    3.32685e+01 0.080011433 0.28669833
## 60   string 1000000 40duplicates    2.09124e+04 2.165506667 6.97509000
##     shell_time  intro_time    tim_time      n2       nlogn
## 1  0.250428000 0.637729000 0.694228667 2.5e+11  6561181.69
## 2  0.607708000 1.459096667 1.508100000 1.0e+12 13815510.56
## 3  0.040877733 0.121494333 0.127097667 1.0e+10  1151292.55
## 4  0.002532720 0.009420033 0.010092207 1.0e+08    92103.40
## 5  0.004941483 0.052645100 0.050111400 2.5e+09   540988.91
## 6  0.018175900 0.055810833 0.058070467 2.5e+09   540988.91
## 7  0.001242187 0.004680827 0.004686100 2.5e+07    42585.97
## 8  0.063239433 0.651658000 0.594141667 2.5e+11  6561181.69
## 9  0.130055000 0.634052000 0.635731333 2.5e+11  6561181.69
## 10 0.001412643 0.009286910 0.009209403 1.0e+08    92103.40
## 11 0.575985667 1.416886667 1.522650000 1.0e+12 13815510.56
## 12 0.591596667 1.446896667 1.515850000 1.0e+12 13815510.56
## 13 0.019270700 0.060891333 0.065351067 2.5e+09   540988.91
## 14 0.000699967 0.005031383 0.004771853 2.5e+07    42585.97
## 15 0.000355217 0.004599507 0.004016070 2.5e+07    42585.97
## 16 0.040603533 0.122850000 0.130229333 1.0e+10  1151292.55
## 17 0.009853843 0.053876167 0.051817133 2.5e+09   540988.91
## 18 0.002845270 0.009706917 0.010166527 1.0e+08    92103.40
## 19 0.268305333 0.691944333 0.730618333 2.5e+11  6561181.69
## 20 0.261900333 0.676314000 0.726678000 2.5e+11  6561181.69
## 21 0.130231333 1.289520000 1.266403333 1.0e+12 13815510.56
## 22 0.001183510 0.004595310 0.004989633 2.5e+07    42585.97
## 23 0.042063733 0.120380000 0.132275333 1.0e+10  1151292.55
## 24 0.018937700 0.059946800 0.061939933 2.5e+09   540988.91
## 25 0.002811417 0.010267833 0.010938233 1.0e+08    92103.40
## 26 0.010340227 0.112963667 0.108361333 1.0e+10  1151292.55
```

```
## 27 0.021917400 0.114444667 0.112808667 1.0e+10  1151292.55
## 28 0.000839413 0.009013073 0.009080243 1.0e+08    92103.40
## 29 0.001266563 0.004846433 0.005275760 2.5e+07    42585.97
## 30 0.281839000 1.369583333 1.389296667 1.0e+12 13815510.56
## 31 0.055238500 0.144711000 0.126737000 2.5e+09   540988.91
## 32 2.264103333 2.072323333 2.283640000 2.5e+11  6561181.69
## 33 0.158814333 0.169828000 0.182254333 2.5e+09   540988.91
## 34 0.022398733 0.027271633 0.031393367 1.0e+08    92103.40
## 35 0.021787300 0.028377600 0.026031267 1.0e+08    92103.40
## 36 0.125729667 0.316126333 0.277936667 1.0e+10  1151292.55
## 37 0.009873377 0.014031200 0.015419933 2.5e+07    42585.97
## 38 2.108266667 1.999353333 1.903406667 2.5e+11  6561181.69
## 39 0.159448667 0.168367667 0.188589333 2.5e+09   540988.91
## 40 2.537876667 2.100453333 2.300713333 2.5e+11  6561181.69
## 41 0.009366907 0.013428133 0.014168700 2.5e+07    42585.97
## 42 0.358975667 0.357869333 0.400953667 1.0e+10  1151292.55
## 43 0.010015960 0.013369767 0.014184433 2.5e+07    42585.97
## 44 0.343863000 0.344172667 0.332346333 1.0e+10  1151292.55
## 45 5.711163333 4.502906667 4.884273333 1.0e+12 13815510.56
## 46 0.023881800 0.032998767 0.031517233 1.0e+08    92103.40
## 47 6.360383333 5.695826667 6.140253333 1.0e+12 13815510.56
## 48 1.633923333 6.158996667 4.364220000 1.0e+12 13815510.56
## 49 2.345153333 2.077396667 2.374253333 2.5e+11  6561181.69
## 50 0.373383667 0.369915333 0.404510333 1.0e+10  1151292.55
## 51 0.008622350 0.012710933 0.012020333 2.5e+07    42585.97
## 52 4.766696667 4.423480000 4.092100000 1.0e+12 13815510.56
## 53 0.004620507 0.013318967 0.010044357 2.5e+07    42585.97
## 54 0.373124667 0.378857667 0.418543333 1.0e+10  1151292.55
## 55 0.022995000 0.026820000 0.031090067 1.0e+08    92103.40
## 56 0.010273337 0.026747100 0.021408333 1.0e+08    92103.40
## 57 0.708828333 1.975380000 1.660640000 2.5e+11  6561181.69
## 58 0.162428000 0.166120000 0.191197667 2.5e+09   540988.91
## 59 0.151066333 0.171476000 0.155545667 2.5e+09   540988.91
## 60 5.461970000 4.622706667 4.945056667 1.0e+12 13815510.56
```

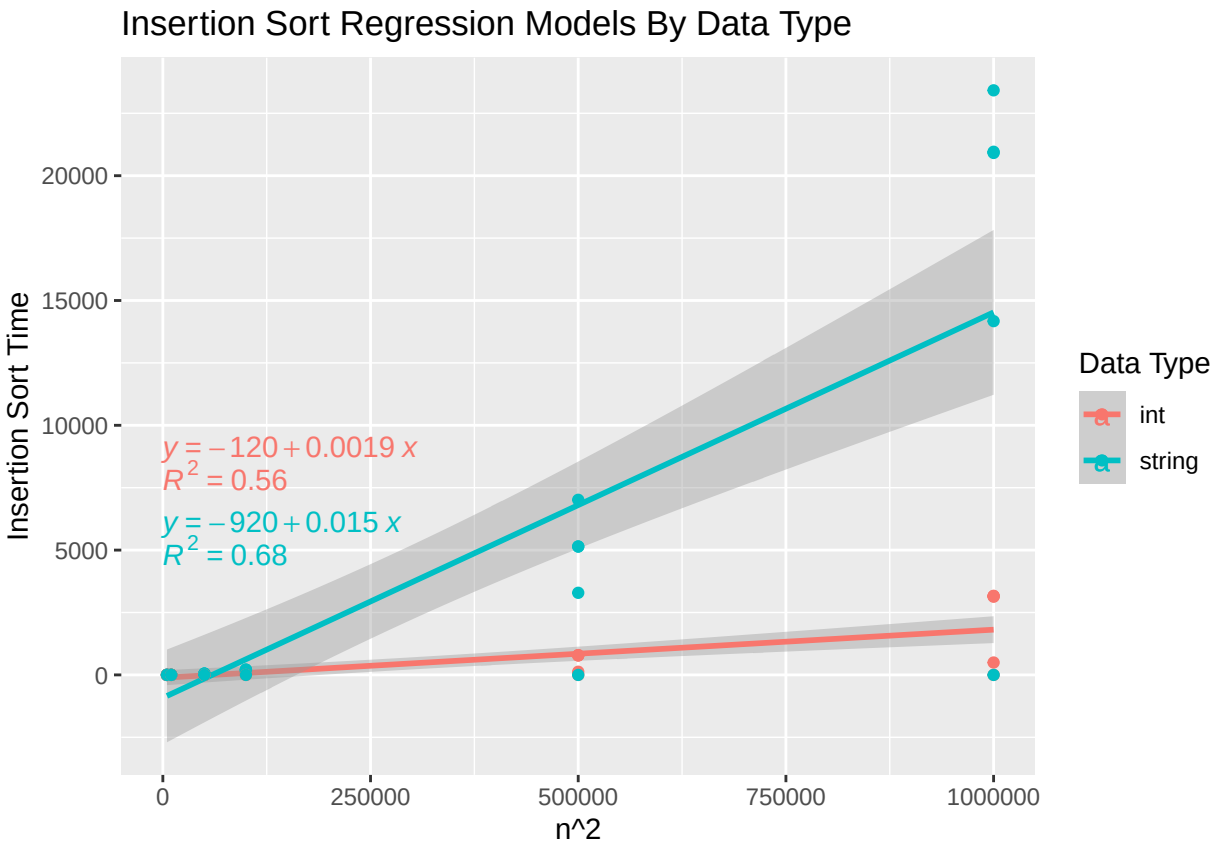
Insertion Sort

```
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"))
```

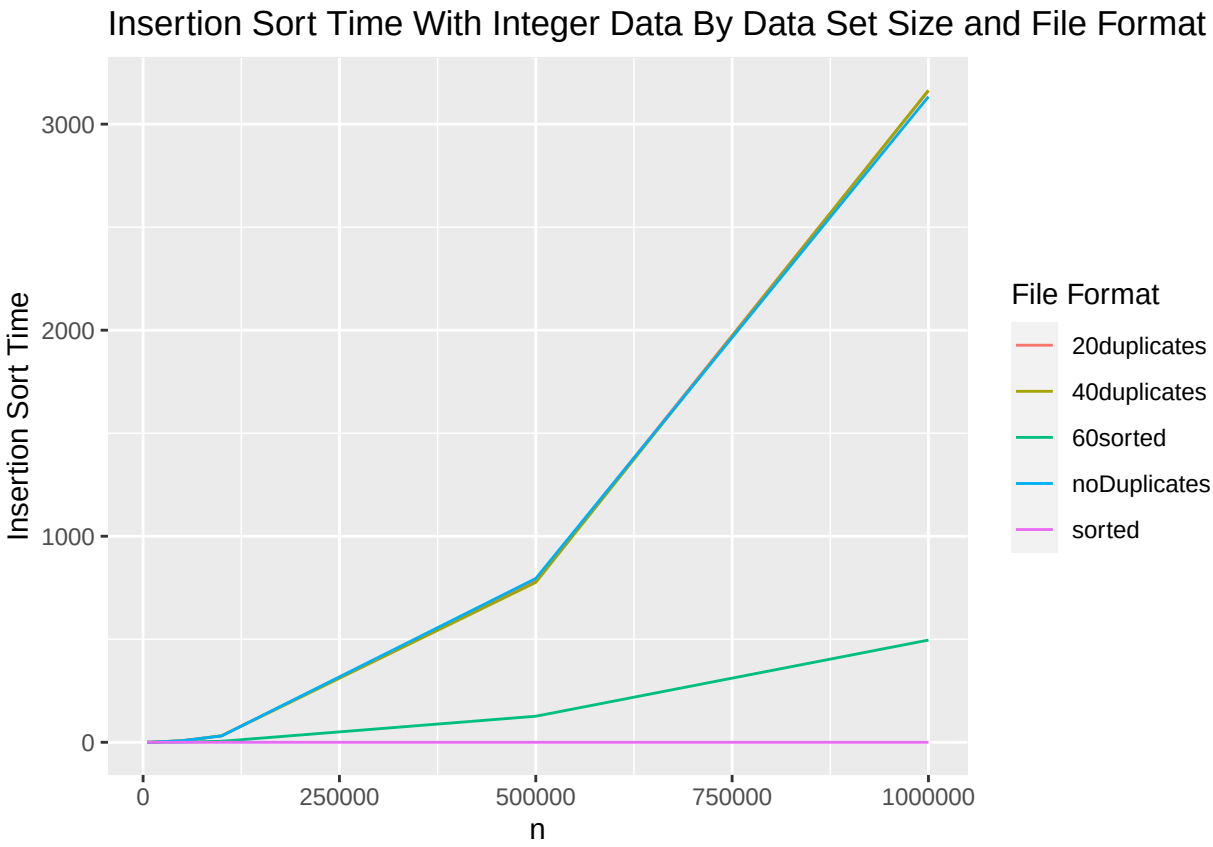



```
ggplot(insertionTimes, aes(x = size, y = insertion_time, color = var_type)) +
  labs(title = "Insertion Sort Regression Models By Data Type", x = "n^2", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(9000, 6000)) +
  stat_cor(aes(label=..rr.label..),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

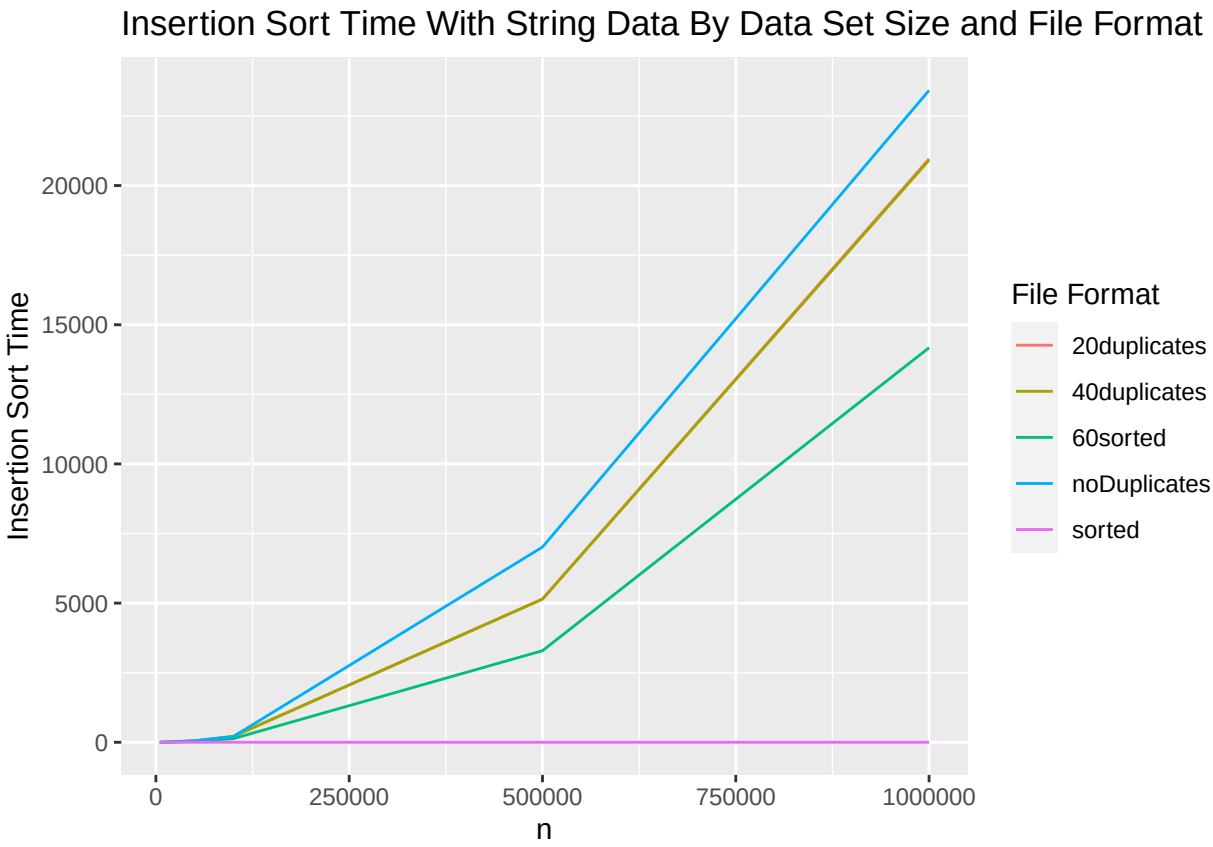
```
## `geom_smooth()` using formula 'y ~ x'
```




```
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"))
```

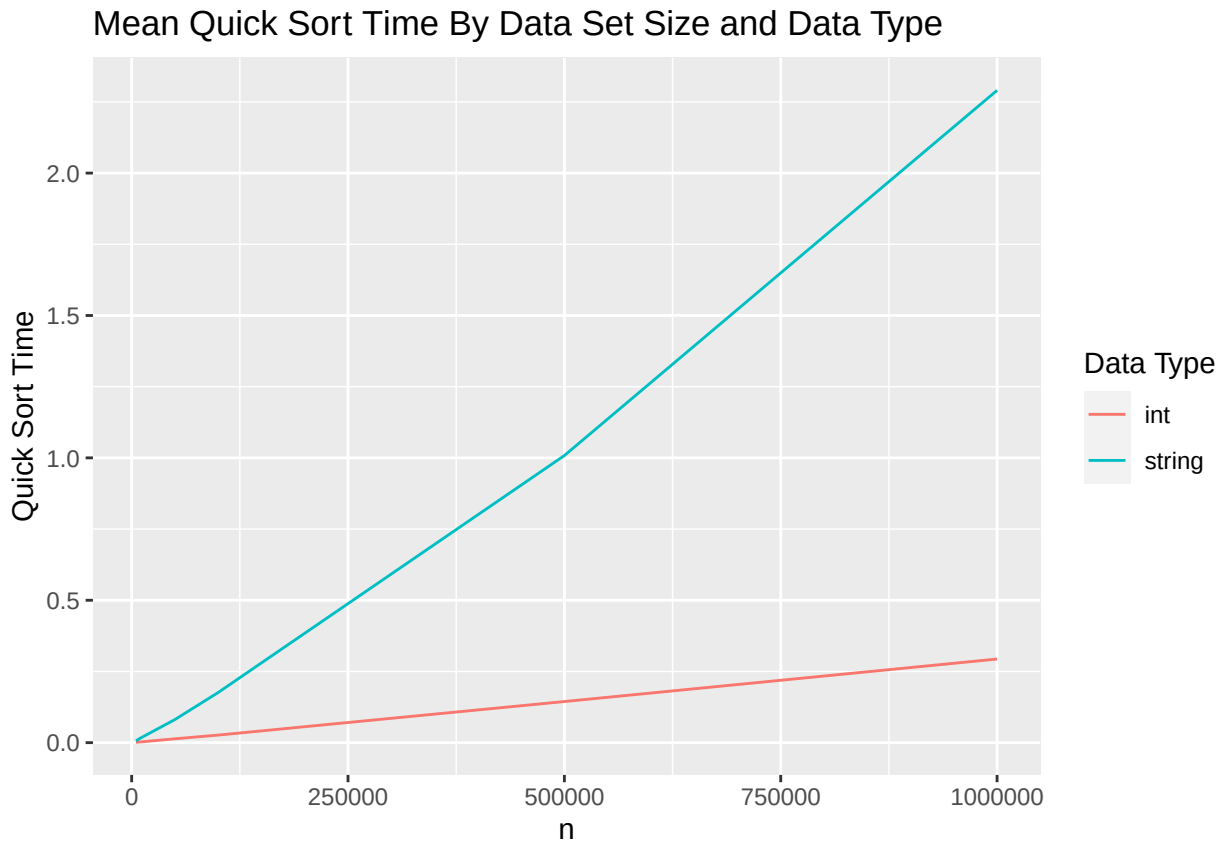



```
insertionStrings = subset(insertionTimes, var_type == "string")
ggplot(insertionStrings, aes(x = size, y = insertion_time, color = format)) +
  geom_line() +
  labs(title = "Insertion Sort Time With String Data By Data Set Size and File Format", x = "n", y = "I
  guides(color = guide_legend(title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



Quick Sort

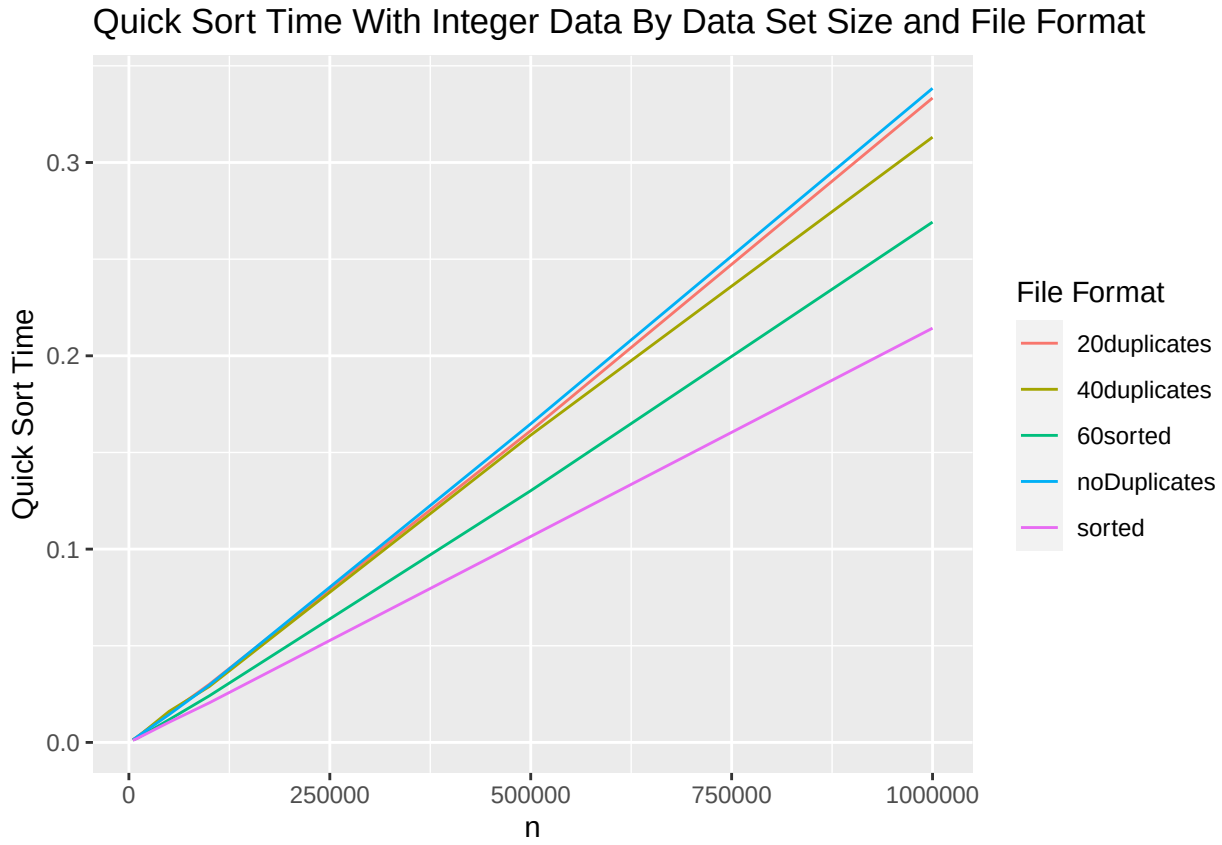
```
quickTimes = aggregate(quick_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
quickTimes2 = aggregate(quick_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(quickTimes2, aes(x = size, y = quick_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Quick Sort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



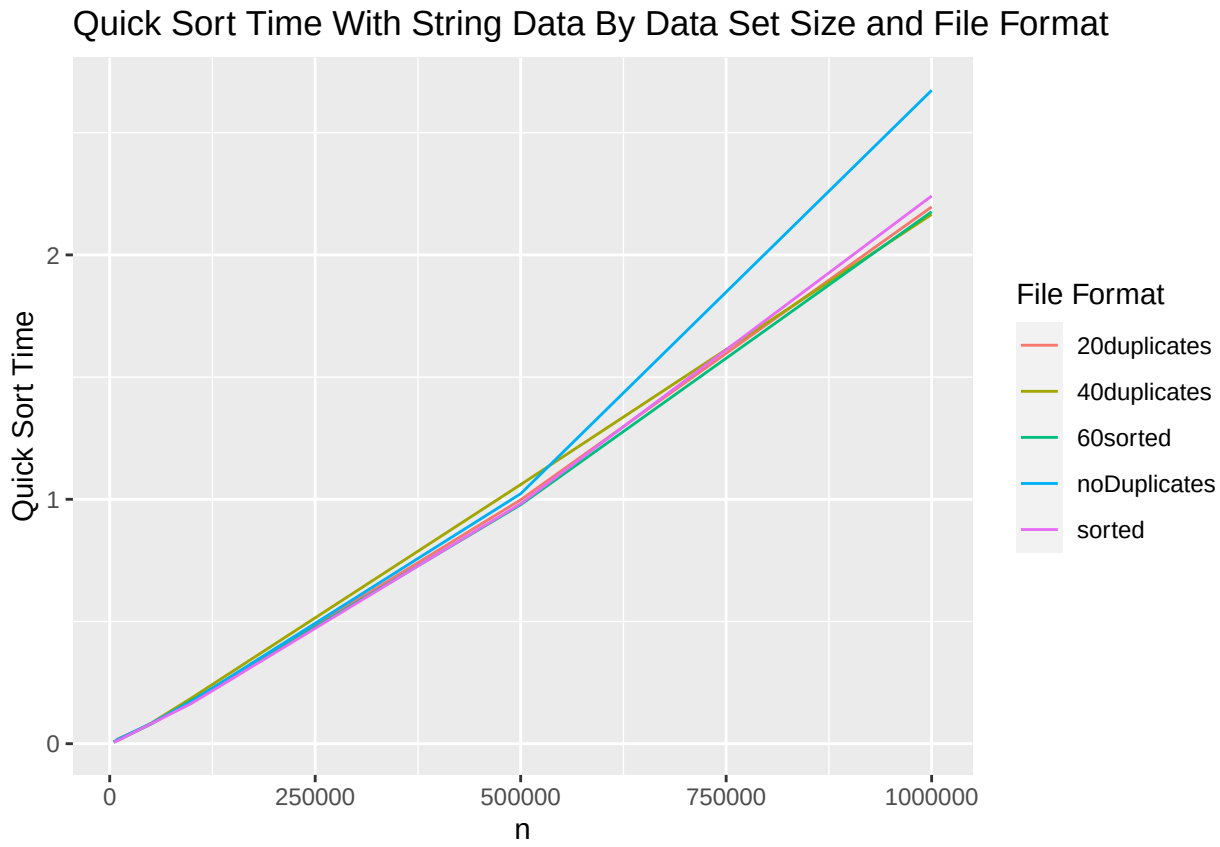

```
ggplot(quickTimes, aes(x = nlogn, y = quick_time, color = var_type)) +
  labs(title = "Quick Sort Regression Models By Data Type", x = "nlogn", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(1.5, 1)) +
  stat_cor(aes(label=..rr.label..),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

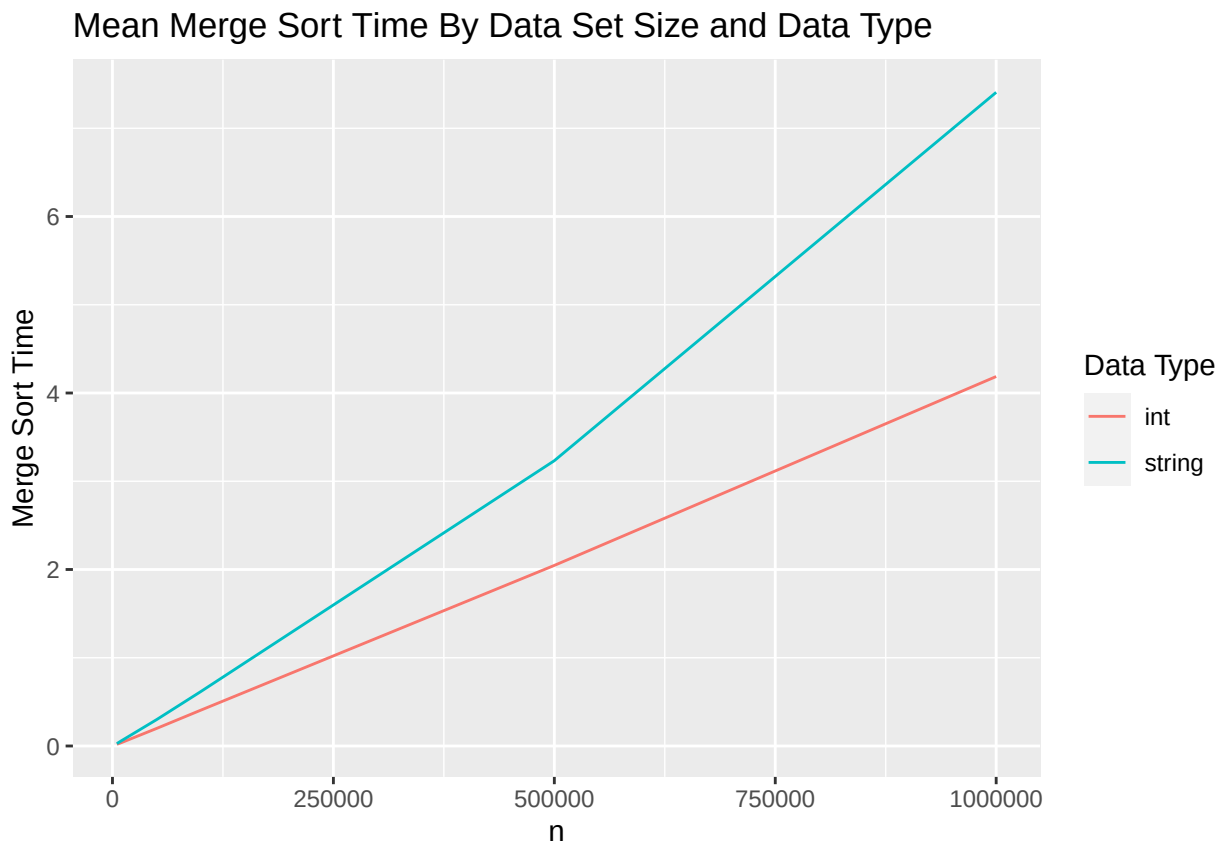



```
quickStrings = subset(quickTimes, var_type == "string")
ggplot(quickStrings, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Quick
  guides(color = guide_legend(title = "File Format"))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



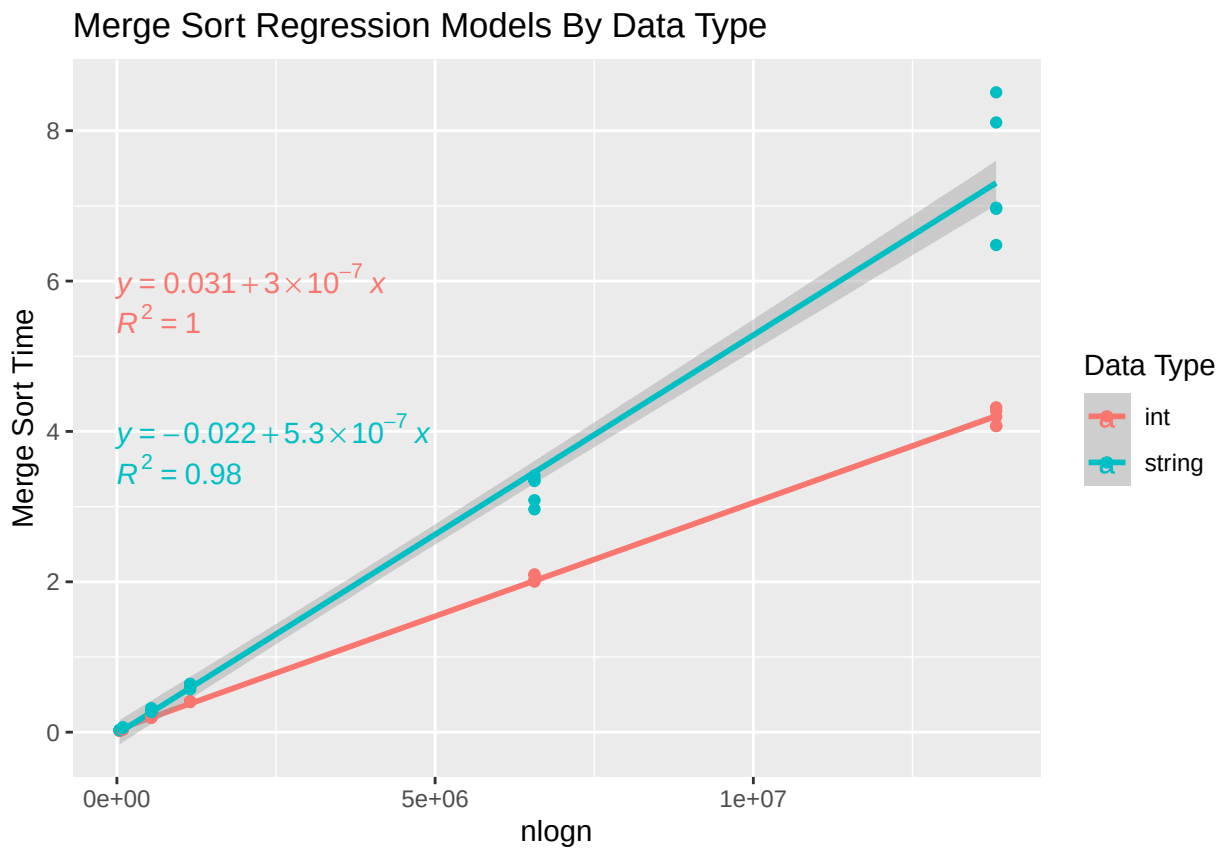
Merge Sort

```
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"))
```

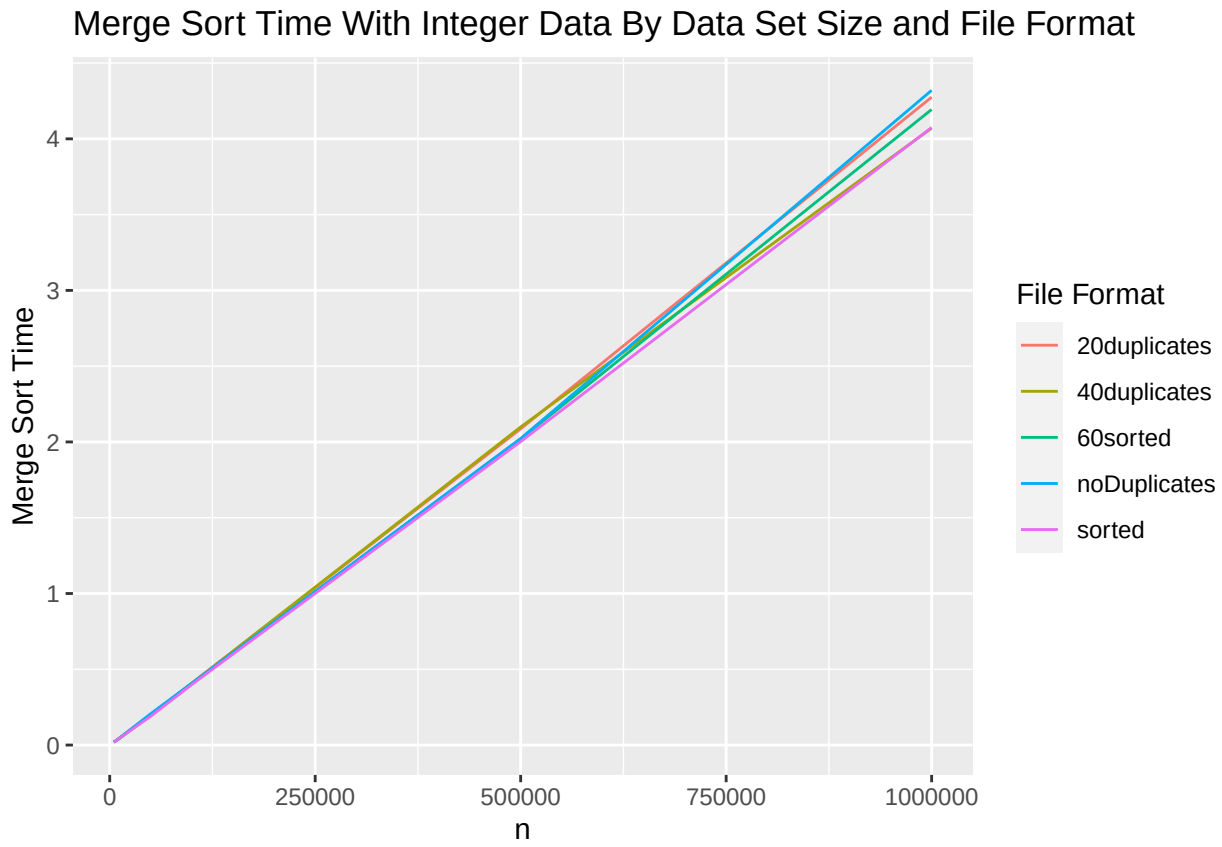



```
ggplot(mergeTimes, aes(x = nlogn, y = merge_time, color = var_type)) +
  labs(title = "Merge Sort Regression Models By Data Type", x = "nlogn", y = "Merge Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(6, 4)) +
  stat_cor(aes(label=..rr.label..),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

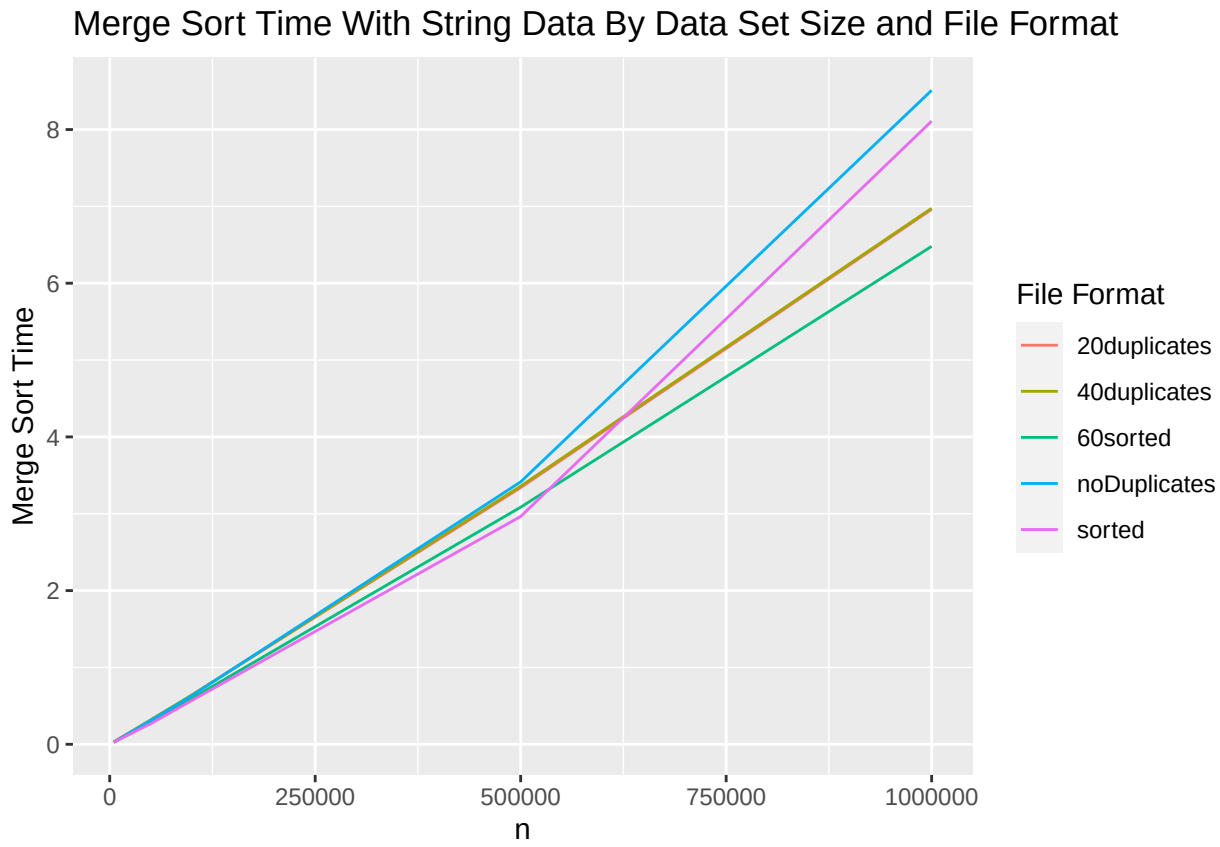
```
## `geom_smooth()` using formula 'y ~ x'
```




```
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 = "Merg
  guides(color = guide_legend(title = "File Format"))
```

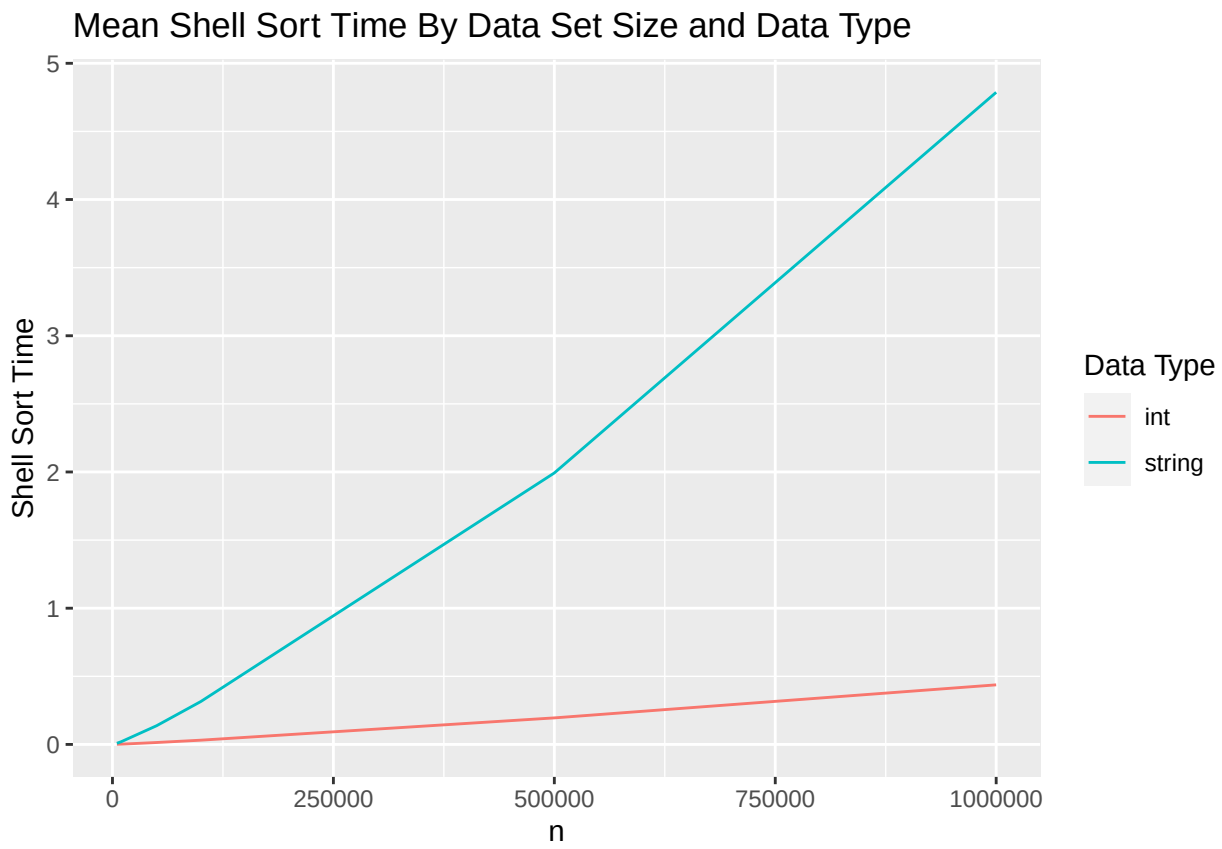



```
mergeStrings = subset(mergeTimes, var_type == "string")
ggplot(mergeStrings, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Merge
  guides(color = guide_legend(title = "File Format"))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



Shell Sort

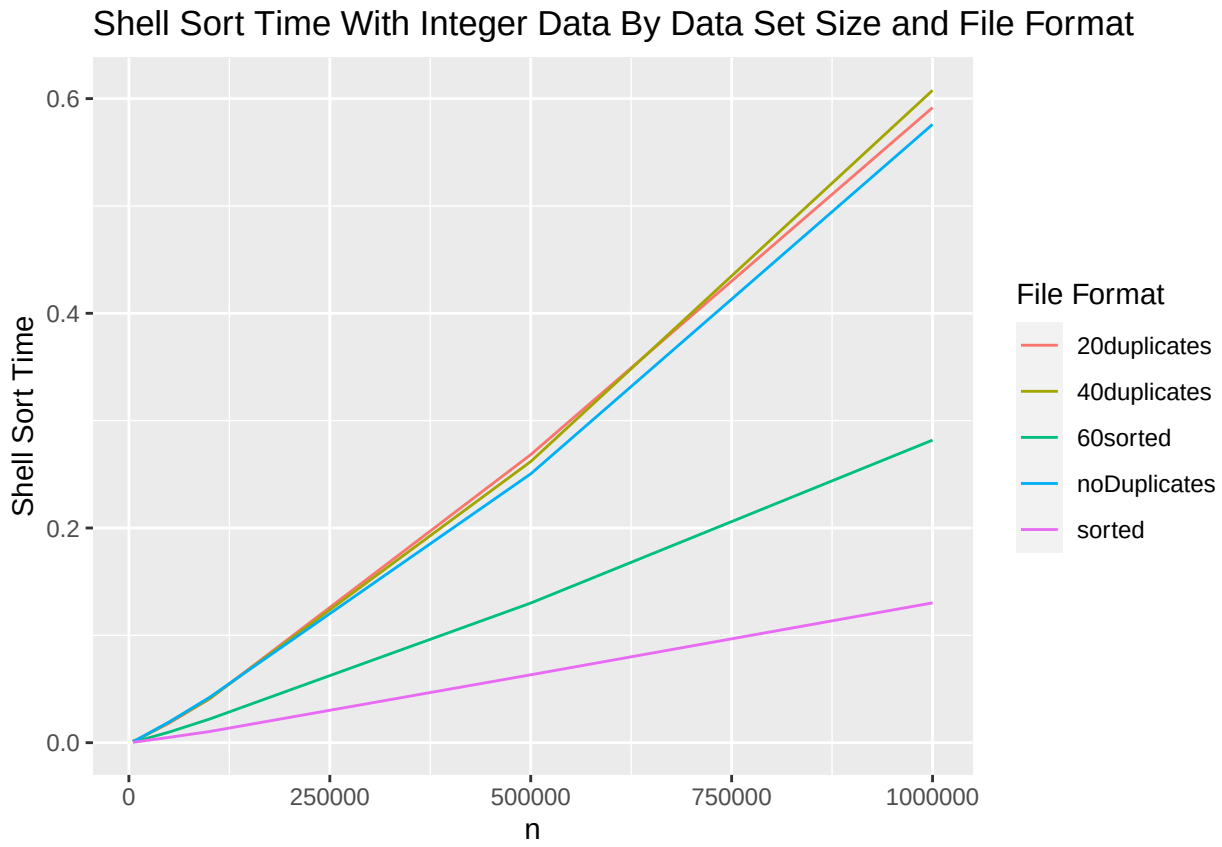
```
shellTimes = aggregate(shell_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
shellTimes2 = aggregate(shell_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(shellTimes2, aes(x = size, y = shell_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Shell Sort 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'
```

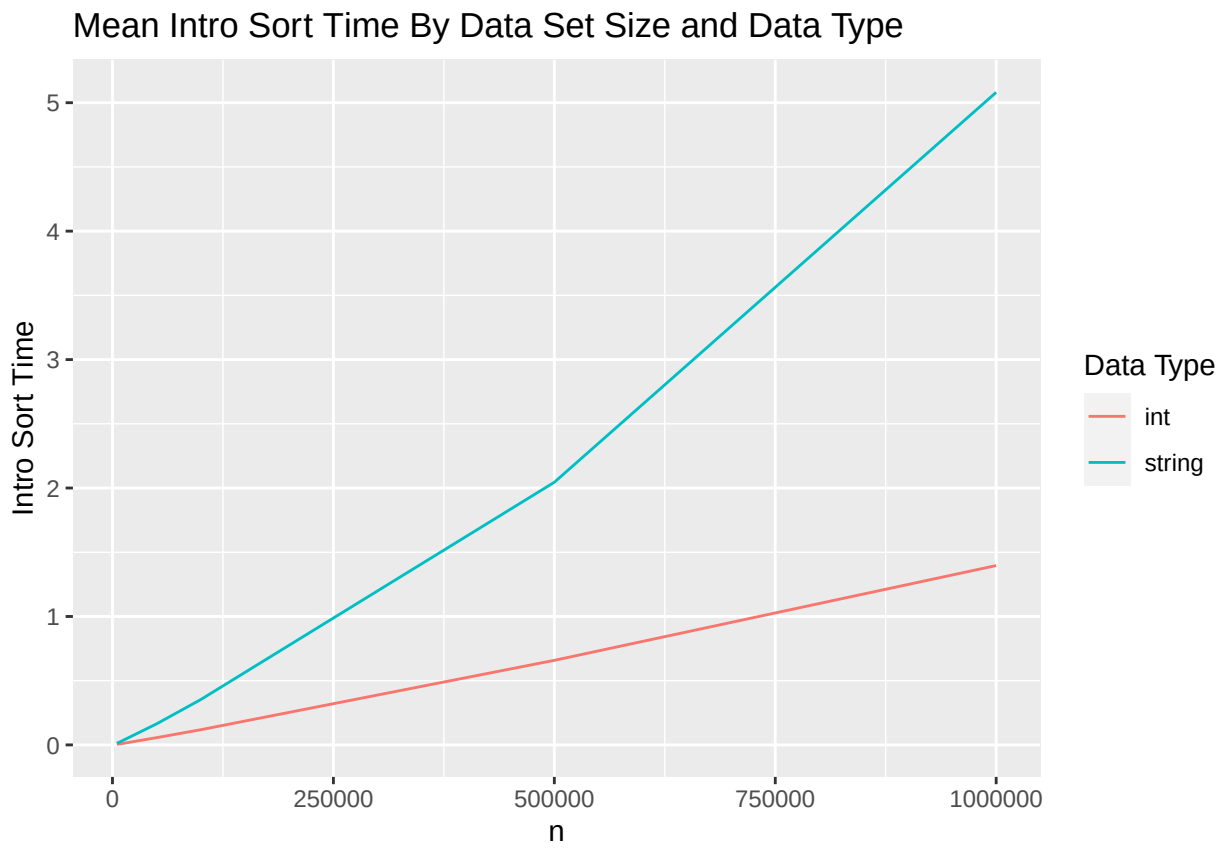

```
shellInts = subset(shellTimes, var_type == "int")
ggplot(shellInts, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Shell
  guides(color = guide_legend(title = "File Format"))
```




```
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"))
```

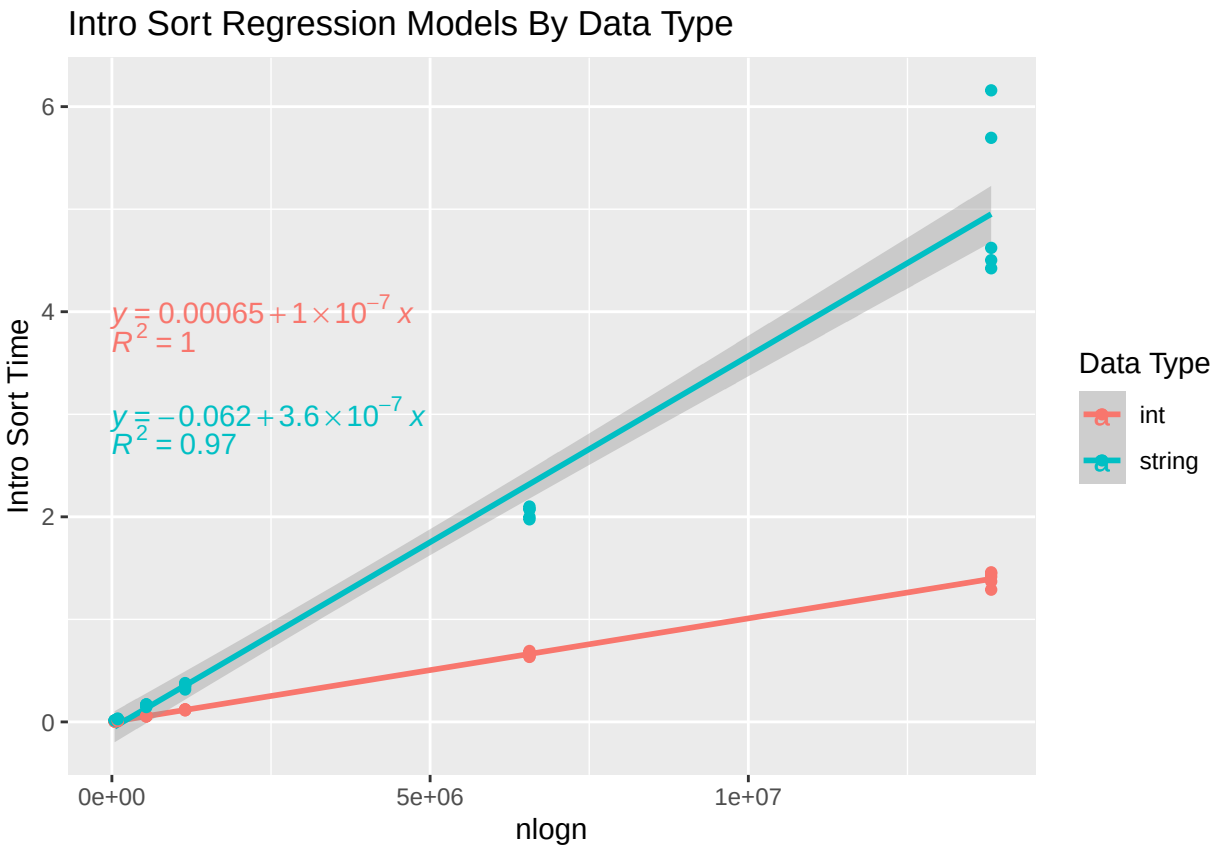
Intro Sort

```
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"))
```

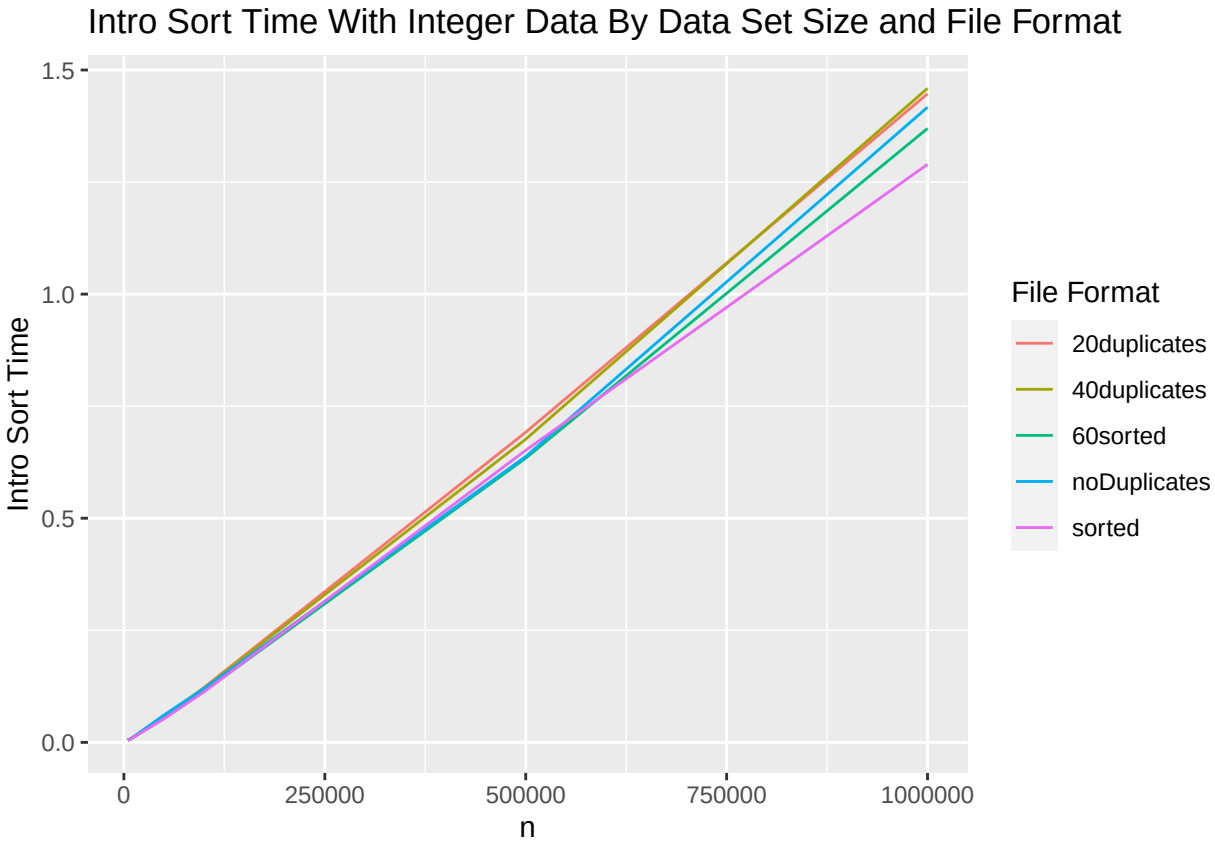



```
ggplot(introTimes, aes(x = nlogn, y = intro_time, color = var_type)) +
  labs(title = "Intro Sort Regression Models By Data Type", x = "nlogn", y = "Intro Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

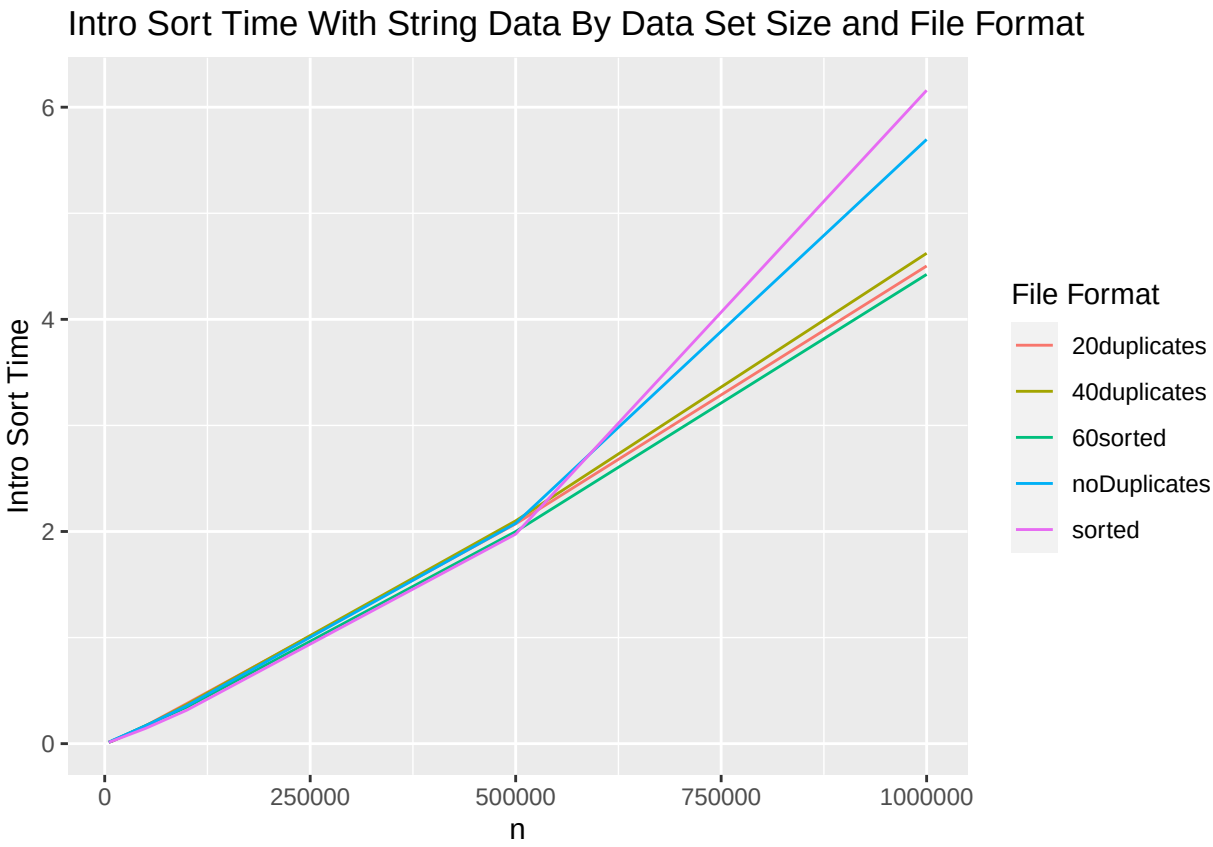
```
## `geom_smooth()` using formula 'y ~ x'
```




```
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"))
```

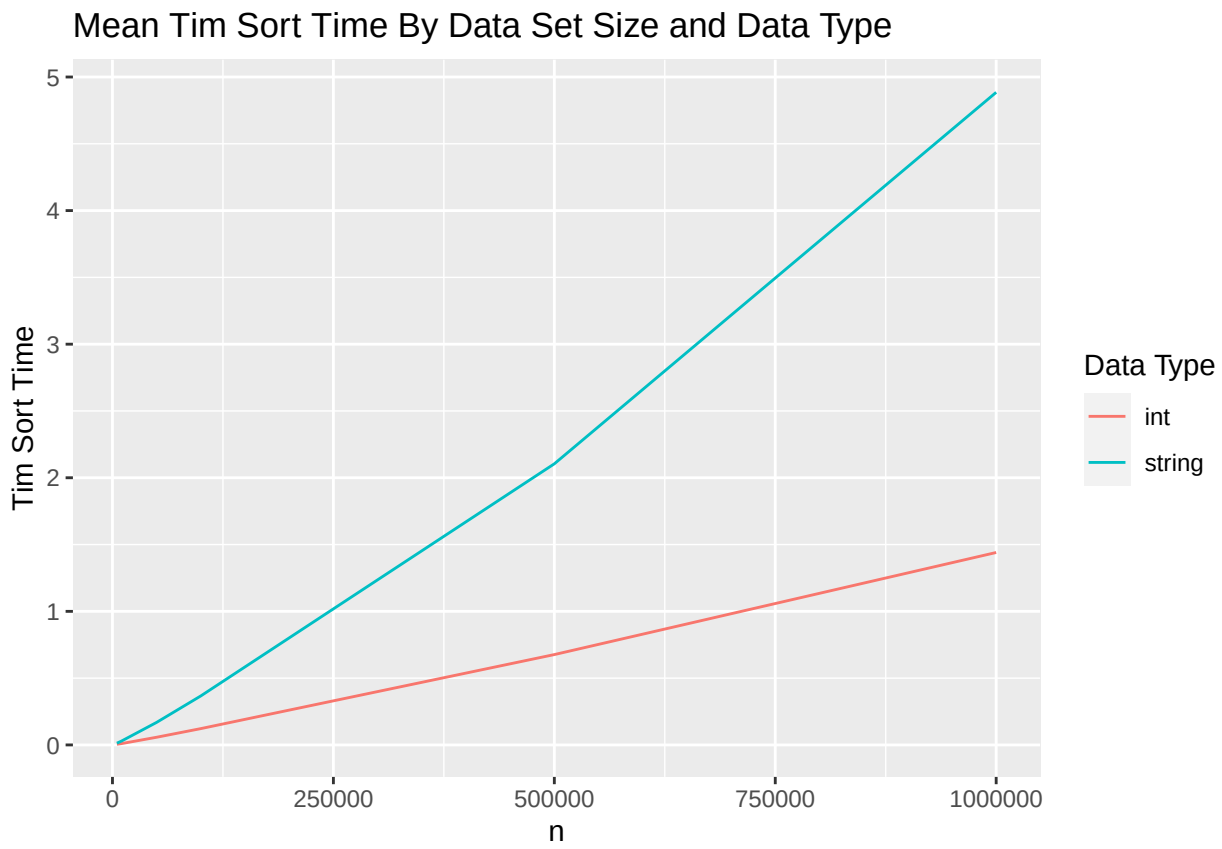



```
introStrings = subset(introTimes, var_type == "string")
ggplot(introStrings, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))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



Tim Sort

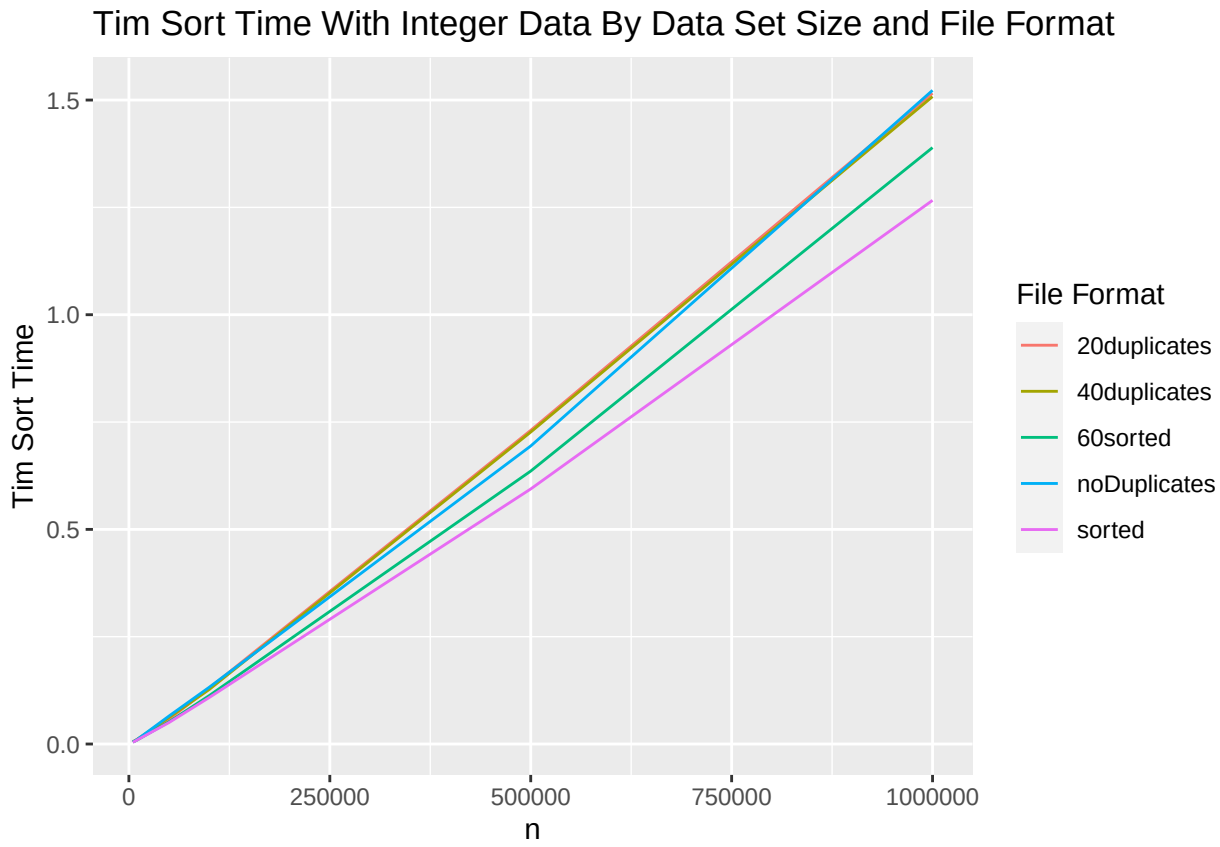
```
timTimes = aggregate(tim_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
timTimes2 = aggregate(tim_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(timTimes2, aes(x = size, y = tim_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Tim Sort Time By Data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



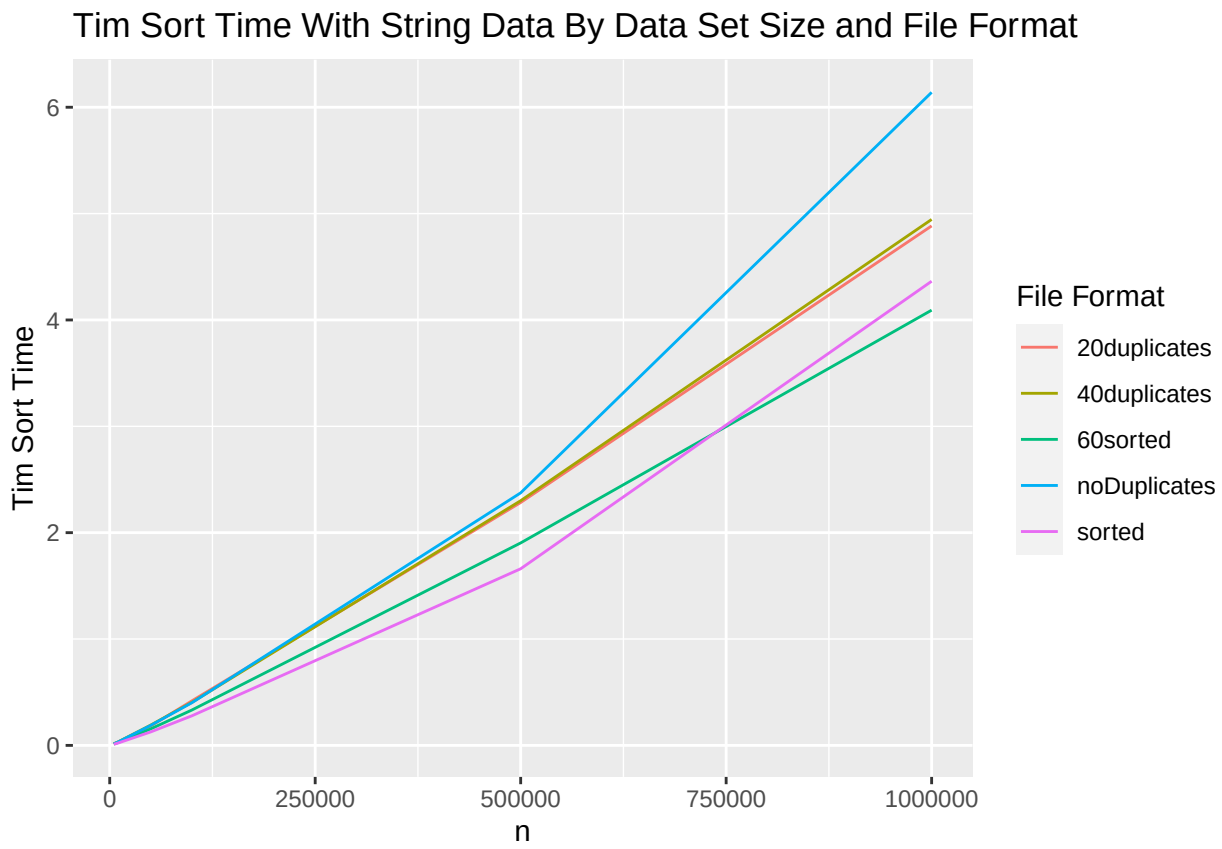

```
ggplot(timTimes, aes(x = nlogn, y = tim_time, color = var_type)) +
  labs(title = "Tim Sort Regression Models By Data Type", x = "nlogn", y = "Tim Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

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


```
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 Sor
  guides(color = guide_legend(title = "File Format"))
```

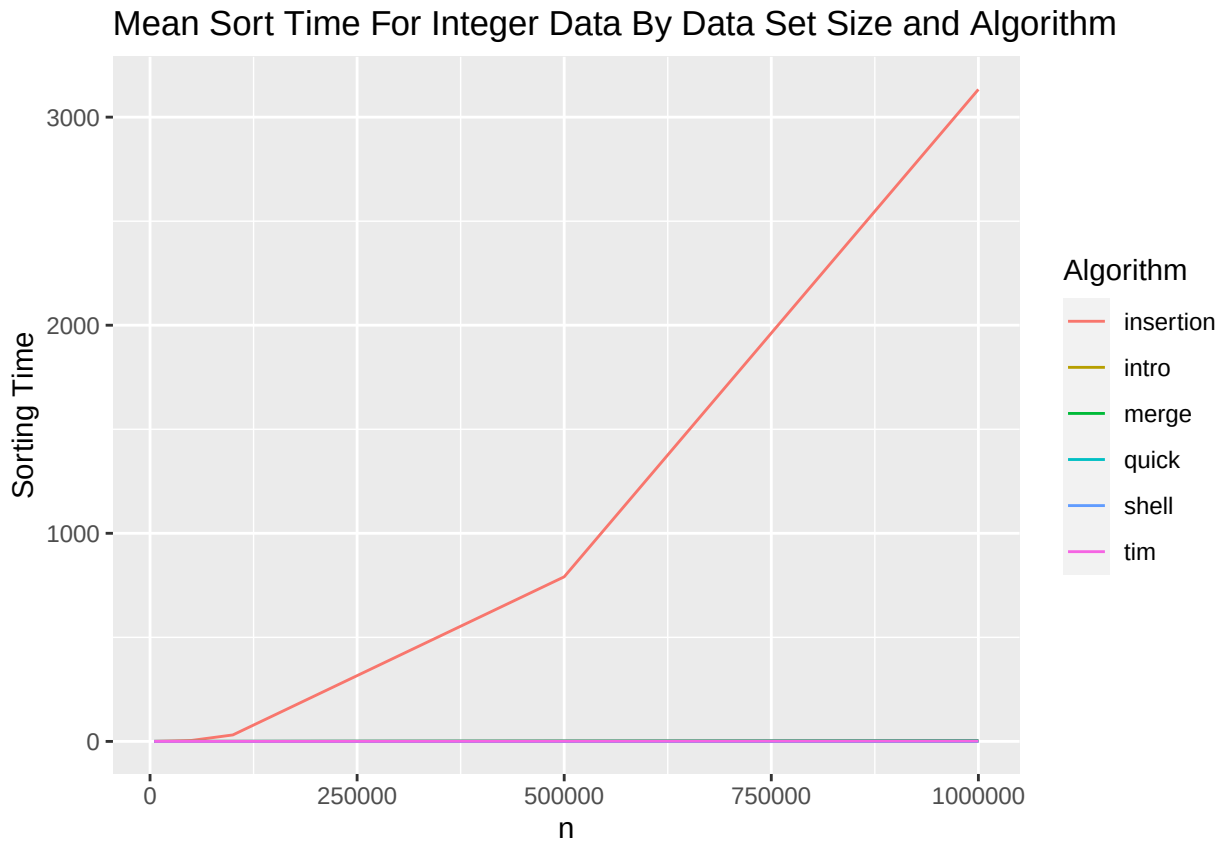


Algorithm Comparison

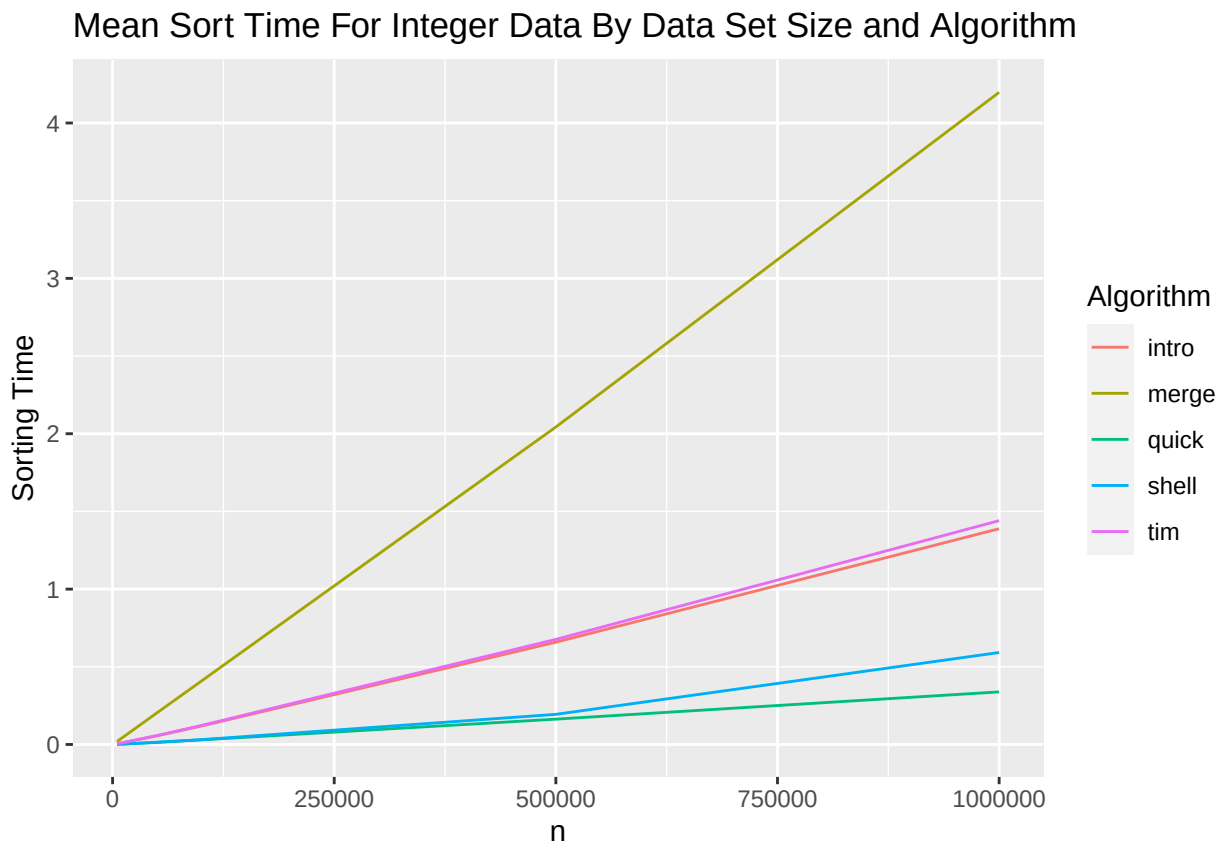
```
data2 = matrix(ncol = 5, nrow = 360)
for (i in 1:6) {
  for (j in 1:60) {
    data2[i * j, 1] = data[j, 1]
    data2[i * j, 2] = data[j, 2]
    data2[i * j, 3] = data[j, 3]
    data2[i * j, 4] = data[j, 3 + i]
    if (i == 1) {
      data2[i * j, 5] = "insertion"
    } else if (i == 2) {
      data2[i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

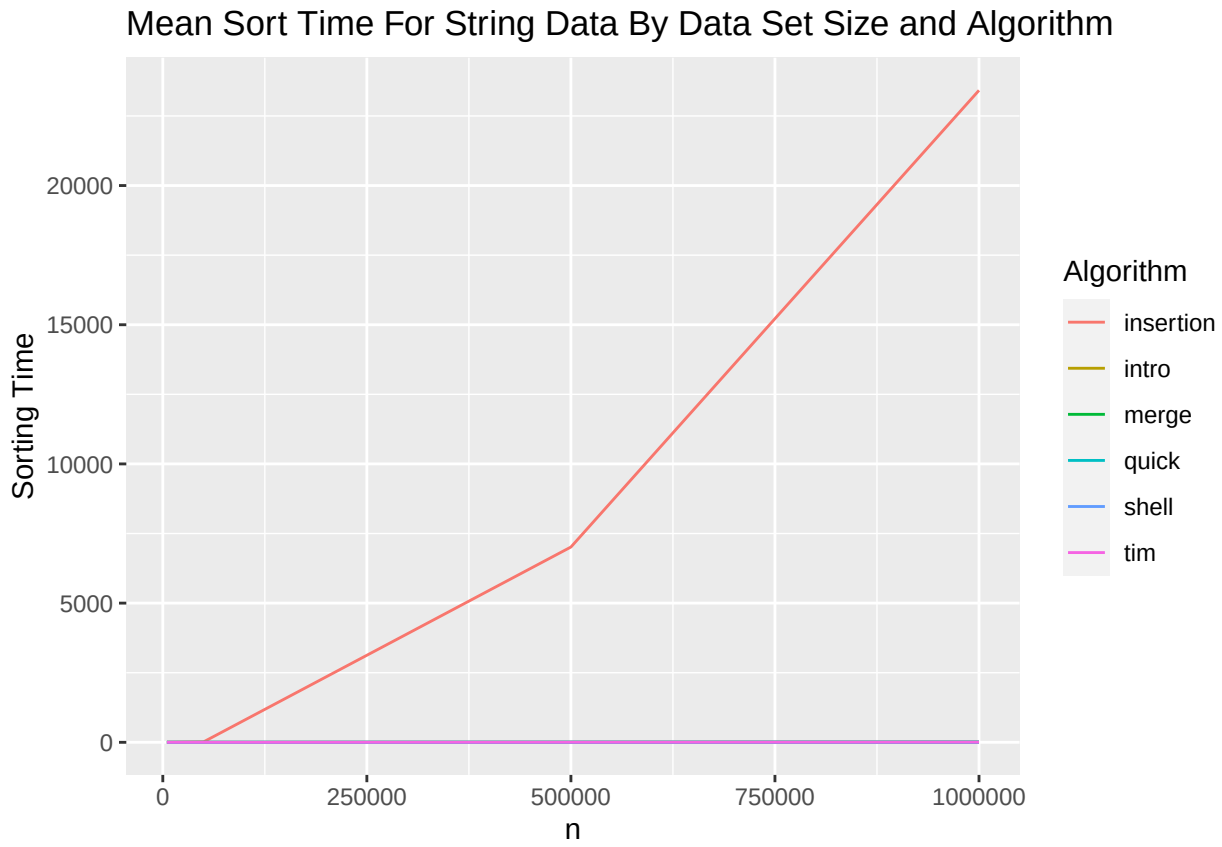
```
integerData = subset(data2, var_type == "int")
integerTimes = aggregate(time ~ algorithm + size, data = integerData, FUN = mean)
ggplot(integerTimes, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x =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



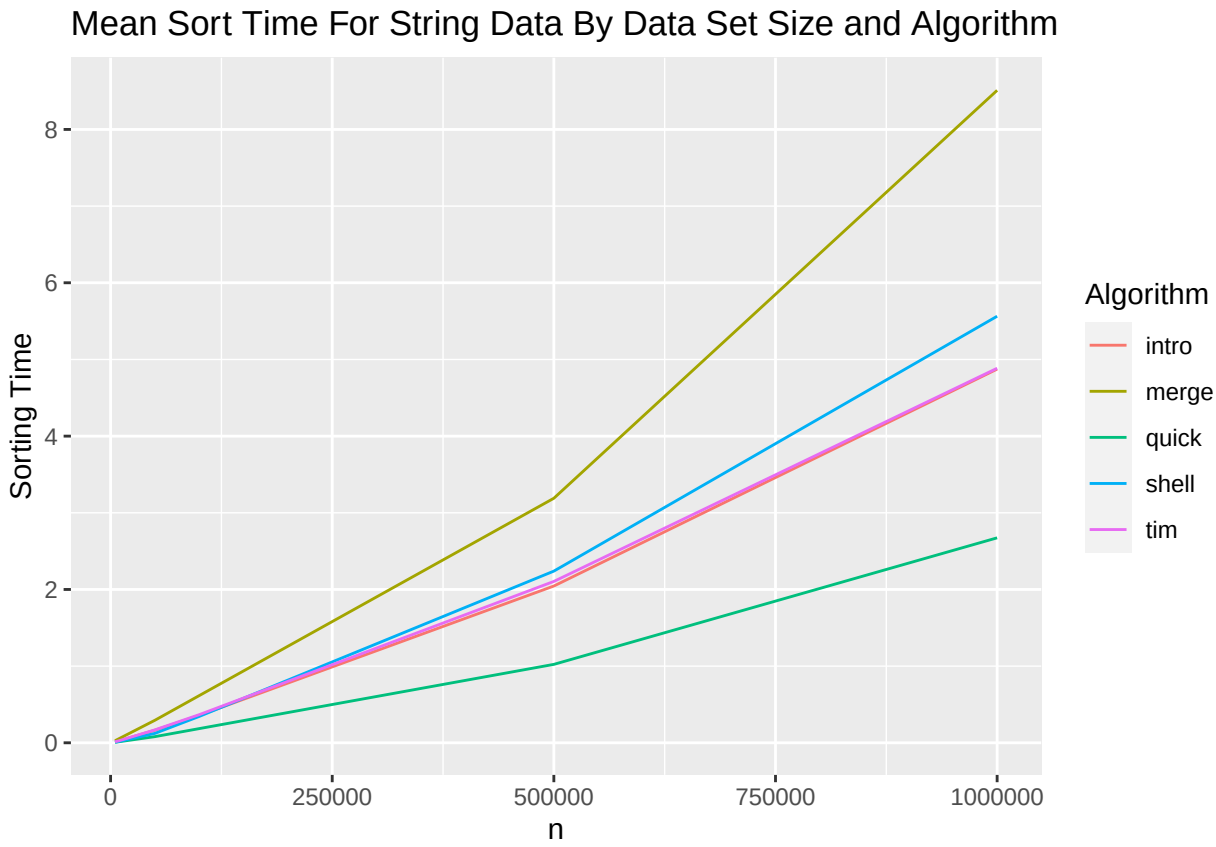

```
integerTimes2 = subset(integerTimes, algorithm != "insertion")
ggplot(integerTimes2, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x = "n", y = "Sorting T
  guides(color = guide_legend(title = "Algorithm"))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




```
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 T
  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 T
  guides(color = guide_legend(title = "Algorithm"))
```


Mean Sort Time For String Data By Data Set Size and Algorithm
Sorting Time
8
6
4
2
0
0
250000
500000
750000
1000000
n
Algorithm
intro
merge
quick
shell
tim