Program 2 Graph Analysis

Ryan Schaefer and Wes Anderson

Create Dataset

```r
library(ggplot2)
library(ggpubr)
data = read.csv("WesDataRun2.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            N/A 0.16311900  2.0440000
## 2        int 1000000 40duplicates            N/A 0.30912300  4.0275800
## 3        int  100000 40duplicates            N/A 0.02848190  0.3987770
## 4        int   10000 40duplicates            N/A 0.00256824  0.0347202
## 5        int   50000       sorted            N/A 0.01190120  0.1809320
## 6        int   50000 20duplicates            N/A 0.01490230  0.1898780
## 7        int    5000 noDuplicates            N/A 0.00137694  0.0193452
## 8        int  500000       sorted            N/A 0.10522300  2.0445900
## 9        int  500000     60sorted            N/A 0.13292200  1.9939200
## 10       int   10000     60sorted            N/A 0.00227108  0.0355995
## 11       int 1000000 noDuplicates            N/A 0.34638300  4.5201700
## 12       int 1000000 20duplicates            N/A 0.33752100  4.2658600
## 13       int   50000 noDuplicates            N/A 0.01441110  0.2050310
## 14       int    5000     60sorted            N/A 0.00127247  0.0221366
## 15       int    5000       sorted            N/A 0.00127496  0.0191041
## 16       int  100000 20duplicates            N/A 0.03029010  0.4087910
## 17       int   50000     60sorted            N/A 0.01202910  0.1934350
## 18       int   10000 noDuplicates            N/A 0.00294144  0.0373266
## 19       int  500000 20duplicates            N/A 0.16059400  2.0761700
## 20       int  500000 40duplicates            N/A 0.15999000  2.1185300
## 21       int 1000000       sorted            N/A 0.21760200  4.0744200
## 22       int    5000 20duplicates            N/A 0.00143460  0.0199234
## 23       int  100000 noDuplicates            N/A 0.02966780  0.4050830
## 24       int   50000 40duplicates            N/A 0.01774390  0.1969860
## 25       int   10000 20duplicates            N/A 0.00269317  0.0389382
## 26       int  100000       sorted            N/A 0.01799210  0.4046140
## 27       int  100000     60sorted            N/A 0.02469870  0.4080430
## 28       int   10000       sorted            N/A 0.00234071  0.0366878
## 29       int    5000 40duplicates            N/A 0.00132507  0.0186269
## 30       int 1000000     60sorted            N/A 0.29896200  4.2269400
## 31    string   50000       sorted            N/A 0.08249620  0.2698010
## 32    string  500000 20duplicates            N/A 1.01595000  3.3300900
## 33    string   50000 20duplicates            N/A 0.08083030  0.3209850
## 34   string   10000 40duplicates            N/A 0.01414800  0.0571073
## 35   string   10000     60sorted            N/A 0.01523460  0.0530800
## 36   string  100000       sorted            N/A 0.16499000  0.5655650
## 37   string    5000 40duplicates            N/A 0.00715901  0.0288678
## 38   string  500000     60sorted            N/A 0.97917300  3.1150900
## 39   string   50000 noDuplicates            N/A 0.08521730  0.3133490
## 40   string  500000 40duplicates            N/A 1.10434000  3.3340200
## 41   string    5000 20duplicates            N/A 0.00732701  0.0277267
## 42   string  100000 noDuplicates            N/A 0.17743600  0.6256320
## 43   string    5000 noDuplicates            N/A 0.00583275  0.0283080
## 44   string  100000     60sorted            N/A 0.17600900  0.5972000
## 45   string 1000000 20duplicates            N/A 2.17273000  6.9262200
## 46   string   10000 noDuplicates            N/A 0.01409820  0.0606856
## 47   string 1000000 noDuplicates            N/A 2.14248000  6.9153000
## 48   string 1000000       sorted            N/A 1.98038000  6.1281100
## 49   string  500000 noDuplicates            N/A 1.00782000  3.3847700
## 50   string  100000 40duplicates            N/A 0.17988400  0.6541840
## 51   string    5000     60sorted            N/A 0.00655066  0.0275832
## 52   string 1000000     60sorted            N/A 2.15932000  6.4592400
## 53   string    5000       sorted            N/A 0.00631719  0.0276873
## 54   string  100000 20duplicates            N/A 0.16833800  0.6395300
## 55   string   10000 20duplicates            N/A 0.01464690  0.0663131
## 56   string   10000       sorted            N/A 0.01283870  0.0562771
## 57   string  500000       sorted            N/A 0.99069000  2.9269700
## 58   string   50000 40duplicates            N/A 0.08137860  0.3320400
## 59   string   50000     60sorted            N/A 0.08249490  0.2832150
## 60   string 1000000 40duplicates            N/A 2.23232000  6.9397400
##      shell_time intro_time   tim_time      n2       nlogn
## 1  0.249655000 0.62253100 0.68434700 2.5e+11  6561181.69
## 2  0.575715000 1.47544000 1.49091000 1.0e+12 13815510.56
## 3  0.040216400 0.11151200 0.12167700 1.0e+10  1151292.55
## 4  0.002526790 0.00925920 0.01005860 1.0e+08    92103.40
## 5  0.004914670 0.05027020 0.04938390 2.5e+09   540988.91
## 6  0.016924900 0.05382430 0.05627430 2.5e+09   540988.91
## 7  0.001117190 0.00472860 0.00455358 2.5e+07    42585.97
## 8  0.060436200 0.65459700 0.59463200 2.5e+11  6561181.69
## 9  0.127787000 0.62076500 0.62893000 2.5e+11  6561181.69
## 10 0.001352090 0.00958218 0.00915712 1.0e+08    92103.40
## 11 0.586460000 1.40551000 1.52609000 1.0e+12 13815510.56
## 12 0.589331000 1.42631000 1.48712000 1.0e+12 13815510.56
## 13 0.018380600 0.06020430 0.07034510 2.5e+09   540988.91
## 14 0.000772749 0.00536396 0.00571790 2.5e+07    42585.97
## 15 0.000347991 0.00419751 0.00381418 2.5e+07    42585.97
## 16 0.038165700 0.12212400 0.13126000 1.0e+10  1151292.55
## 17 0.009624630 0.05344460 0.05216700 2.5e+09   540988.91
## 18 0.002732070 0.00903810 0.00993439 1.0e+08    92103.40
## 19 0.262177000 0.68857600 0.73128100 2.5e+11  6561181.69
## 20 0.265510000 0.67221900 0.74552400 2.5e+11  6561181.69
## 21 0.127704000 1.26321000 1.25367000 1.0e+12 13815510.56
## 22 0.001232650 0.00460063 0.00517563 2.5e+07    42585.97
## 23 0.038536700 0.11770200 0.12467700 1.0e+10  1151292.55
## 24 0.019294600 0.05649020 0.05803740 2.5e+09   540988.91
## 25 0.002690530 0.01016440 0.01081900 1.0e+08    92103.40
## 26 0.011339300 0.11849000 0.11307100 1.0e+10  1151292.55
## 27 0.020970300 0.11953500 0.11499900 1.0e+10  1151292.55
## 28 0.000774877 0.00924054 0.00885194 1.0e+08    92103.40
## 29 0.001189500 0.00459657 0.00508247 2.5e+07    42585.97
## 30 0.286935000 1.35392000 1.37602000 1.0e+12 13815510.56
## 31 0.053387700 0.15619300 0.12777700 2.5e+09   540988.91
## 32 2.296730000 2.10495000 2.29102000 2.5e+11  6561181.69
## 33 0.154035000 0.17874400 0.18449700 2.5e+09   540988.91
## 34 0.022948500 0.02874530 0.03120600 1.0e+08    92103.40
## 35 0.020411100 0.03066940 0.02582140 1.0e+08    92103.40
## 36 0.126209000 0.30604100 0.28119700 1.0e+10  1151292.55
## 37 0.010816900 0.01610480 0.01707160 2.5e+07    42585.97
## 38 2.088500000 2.02440000 1.87648000 2.5e+11  6561181.69
## 39 0.157063000 0.17130500 0.19053100 2.5e+09   540988.91
## 40 2.511580000 2.06044000 2.27695000 2.5e+11  6561181.69
## 41 0.009822830 0.01373760 0.01367700 2.5e+07    42585.97
## 42 0.347116000 0.37610000 0.40498700 1.0e+10  1151292.55
## 43 0.010249800 0.01284950 0.01533420 2.5e+07    42585.97
## 44 0.347674000 0.35303400 0.33707500 1.0e+10  1151292.55
## 45 5.638070000 4.24939000 4.84042000 1.0e+12 13815510.56
## 46 0.022407600 0.03111430 0.03051900 1.0e+08    92103.40
## 47 5.475730000 4.35755000 4.88015000 1.0e+12 13815510.56
## 48 1.511510000 3.98404000 3.48350000 1.0e+12 13815510.56
## 49 2.329970000 2.07662000 2.31346000 2.5e+11  6561181.69
## 50 0.379005000 0.37124300 0.39752800 1.0e+10  1151292.55
## 51 0.008085880 0.01261320 0.01142550 2.5e+07    42585.97
## 52 4.723270000 4.27612000 4.01166000 1.0e+12 13815510.56
## 53 0.004755980 0.01354640 0.00992384 2.5e+07    42585.97
## 54 0.371193000 0.37324700 0.39010500 1.0e+10  1151292.55
## 55 0.026781100 0.02543590 0.02956170 1.0e+08    92103.40
## 56 0.011275100 0.02696940 0.02361330 1.0e+08    92103.40
## 57 0.697953000 1.93084000 1.59814000 2.5e+11  6561181.69
## 58 0.161443000 0.16335500 0.18766500 2.5e+09   540988.91
## 59 0.146549000 0.18255400 0.15197100 2.5e+09   540988.91
## 60 5.439670000 4.62699000 4.87112000 1.0e+12 13815510.56
```

Insertion Sort

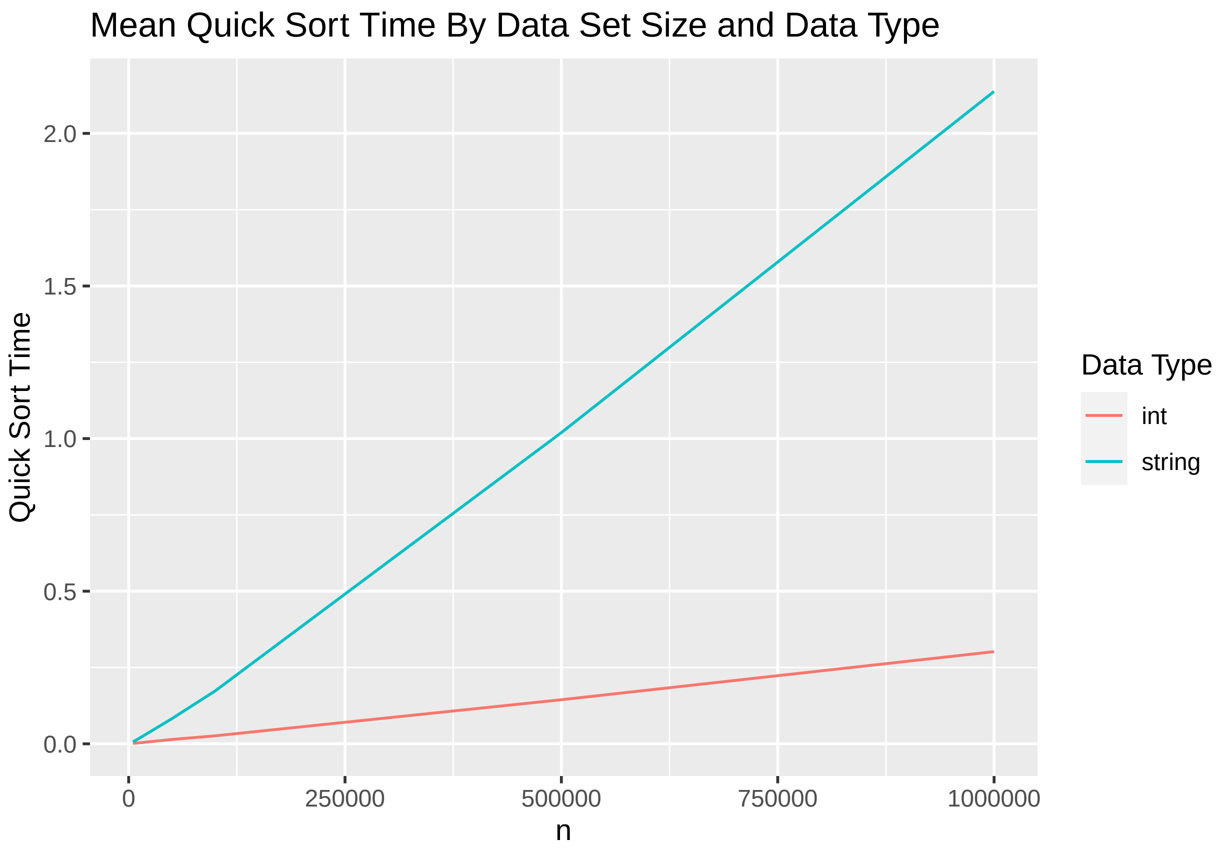
```r
# insertionTimes = aggregate(insertion_time ~ var_type + size + n2 + format, data = data, FUN = mean)
# insertionTimes2 = aggregate(insertion_time ~ var_type + size + n2, data = data, FUN = mean)
# ggplot(insertionTimes2, aes(x = size, y = insertion_time, color = var_type)) +
#   geom_line() +
#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

```r
# 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"))
# 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 =
#   guides(color = guide_legend(title = "File Format"))
```

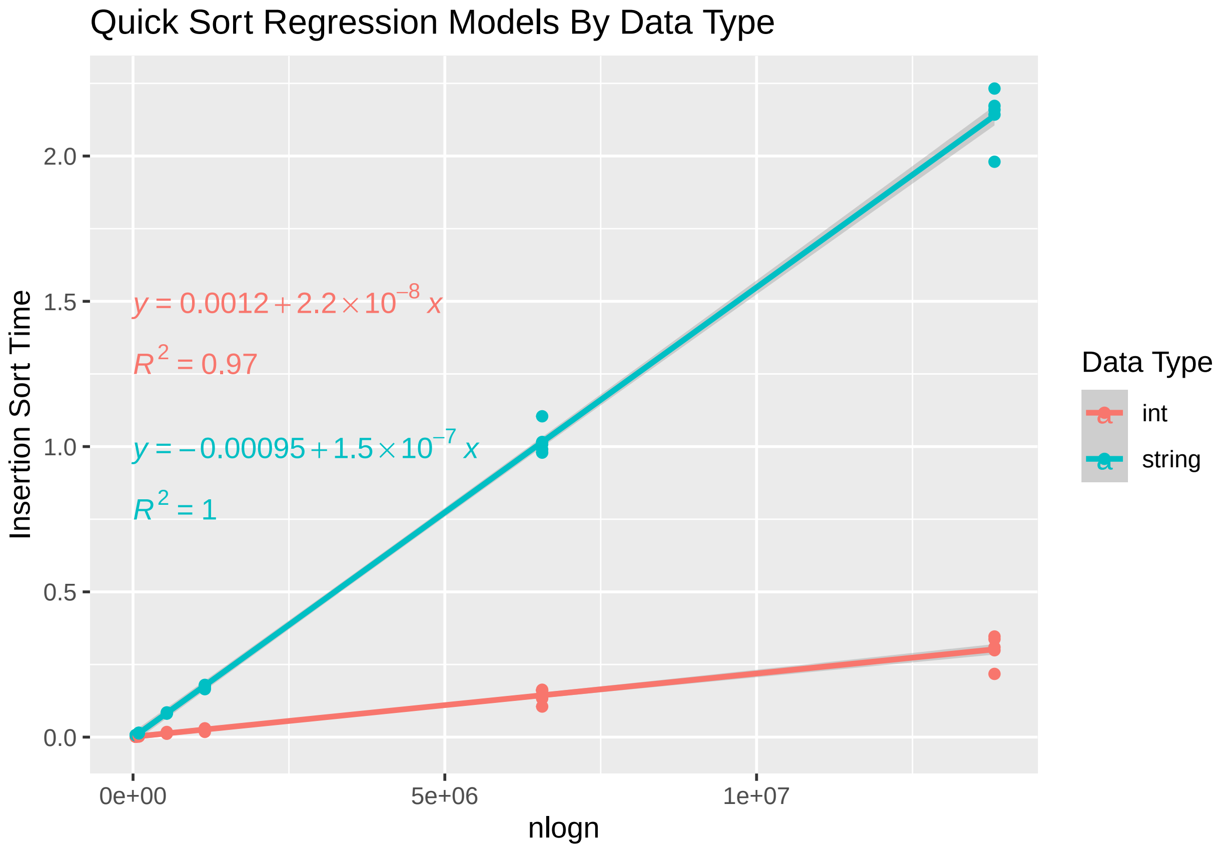
Quick Sort

```
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"))
```

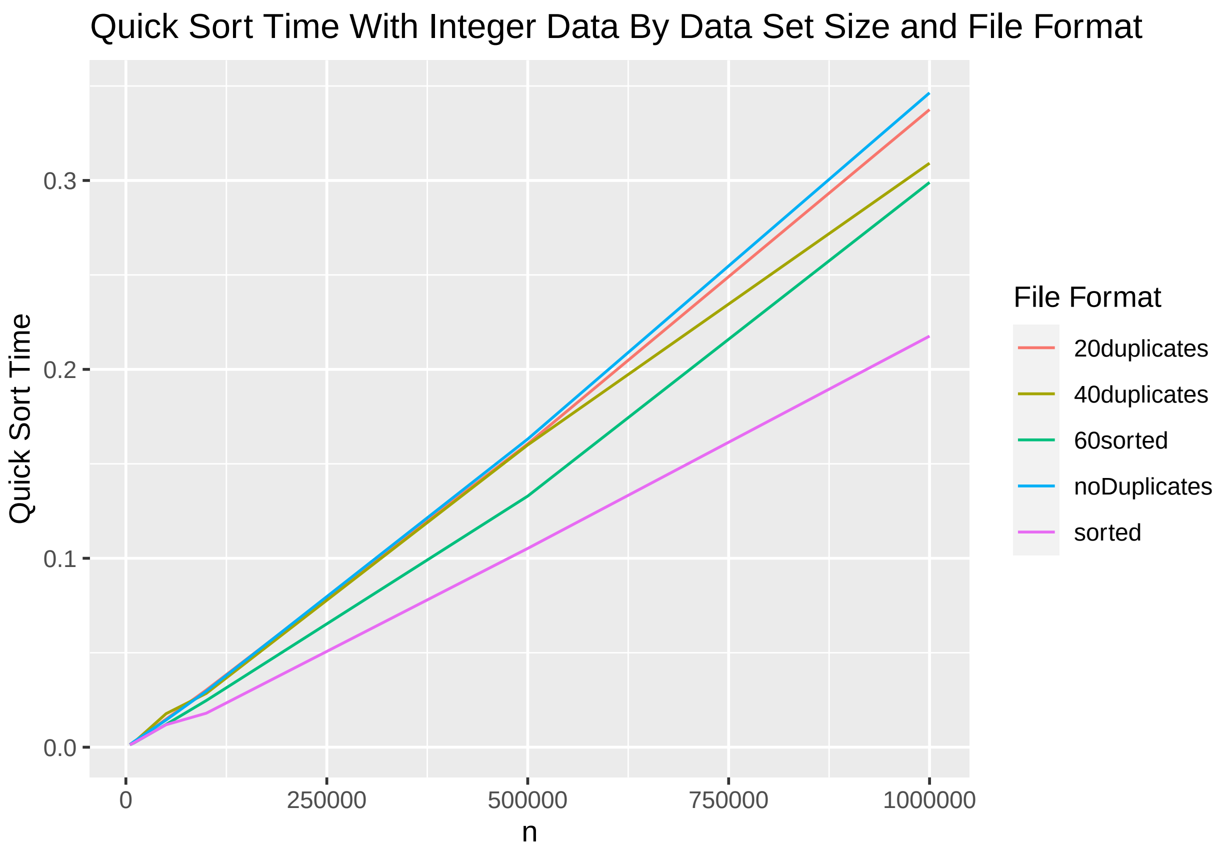


```
ggplot(quickTimes, aes(x = nlogn, y = quick_time, color = var_type)) +
  labs(title = "Quick Sort Regression Models By Data Type", x = "nlogn", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(1.5, 1)) +
  stat_cor(aes(label=..rr.label..),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

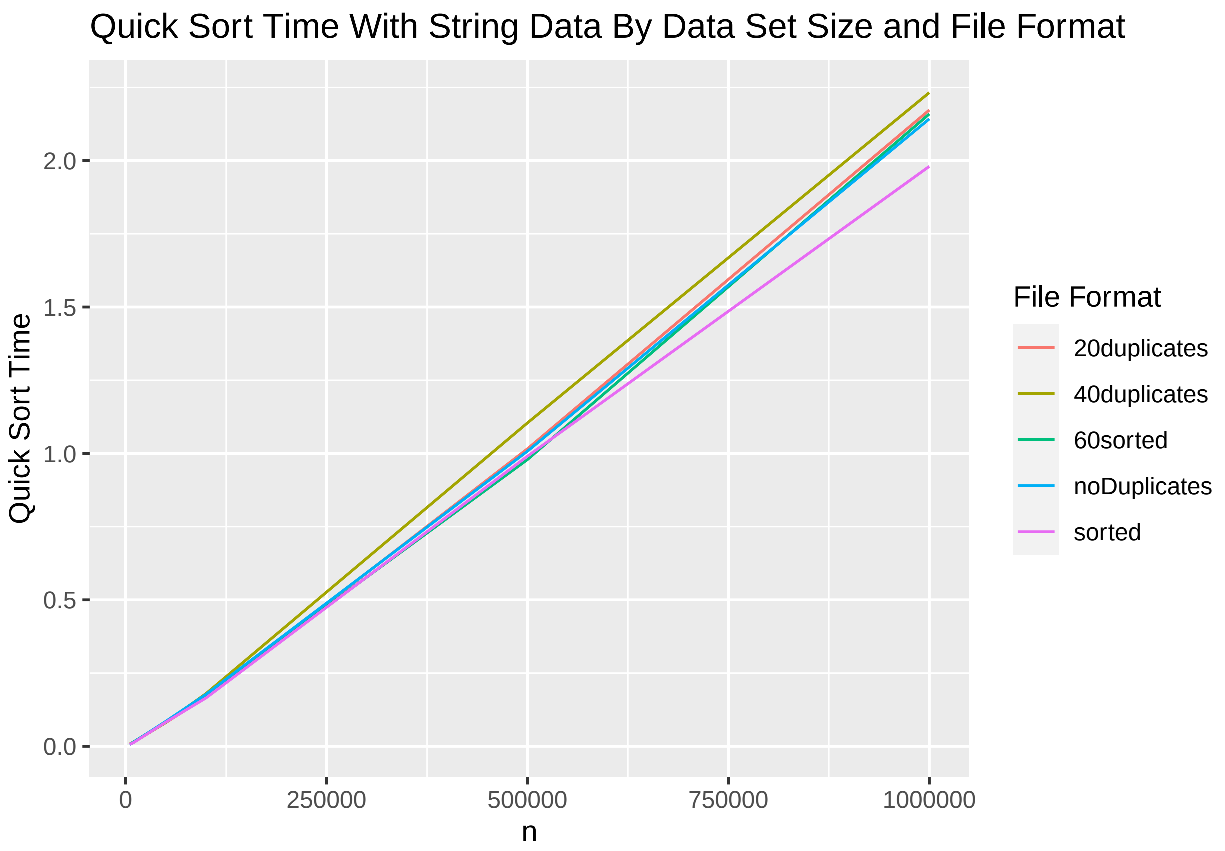
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

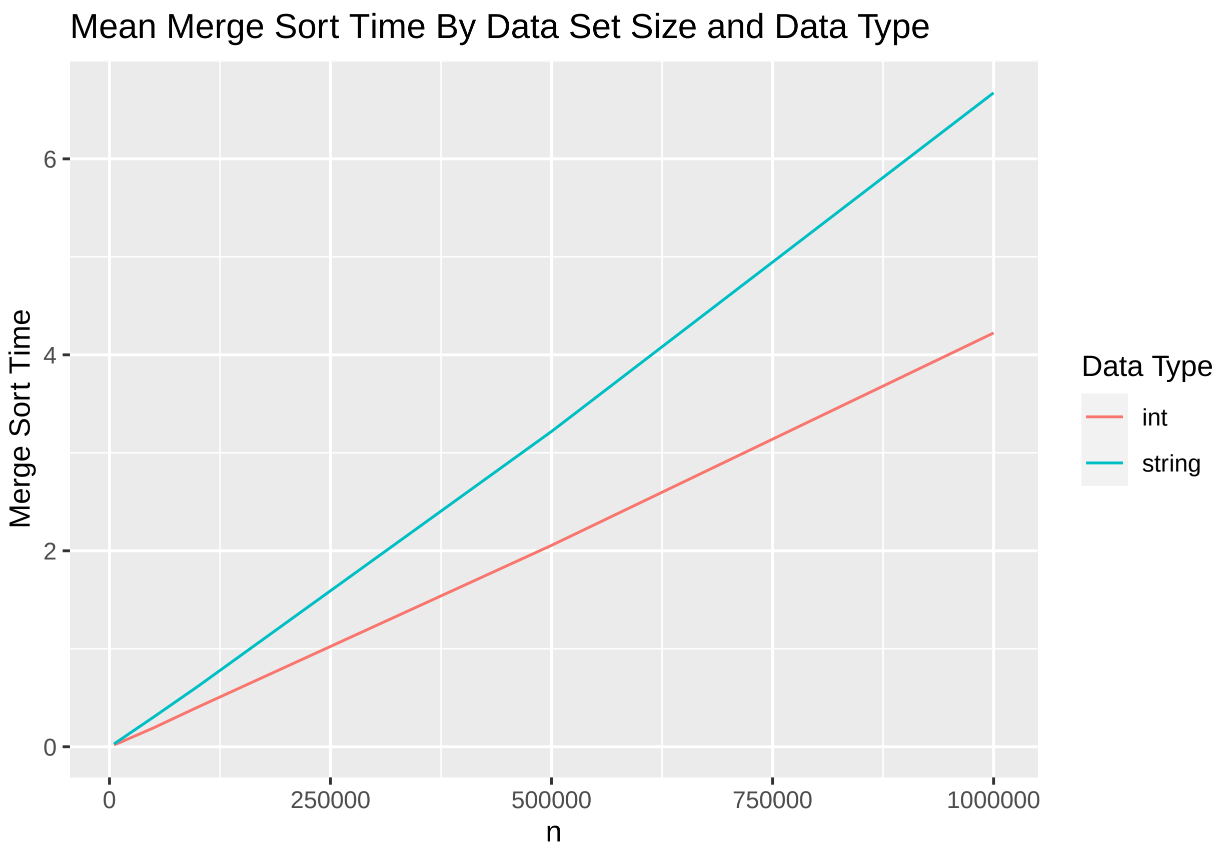


```
quickStrings = subset(quickTimes, var_type == "string")
ggplot(quickStrings, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Quick
  guides(color = guide_legend(title = "File Format"))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



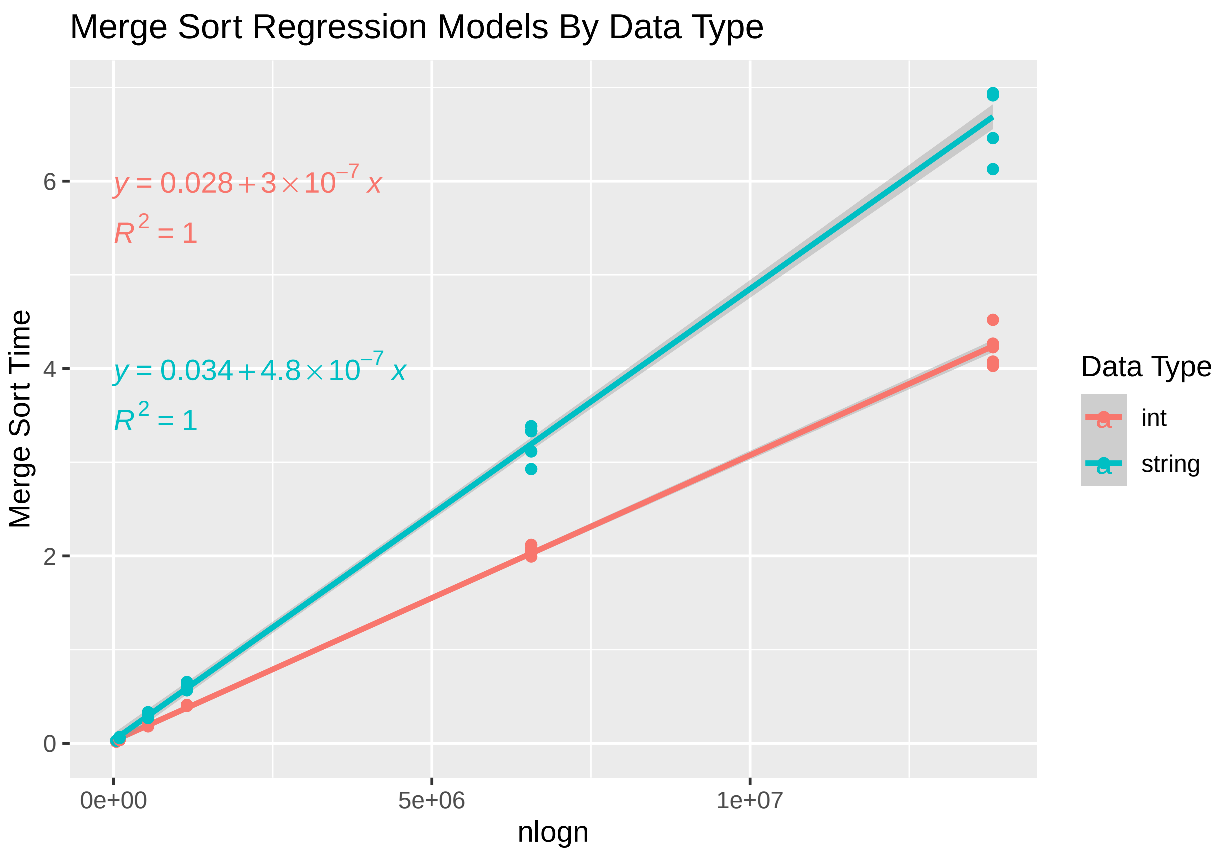
Merge Sort

```
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"))
```

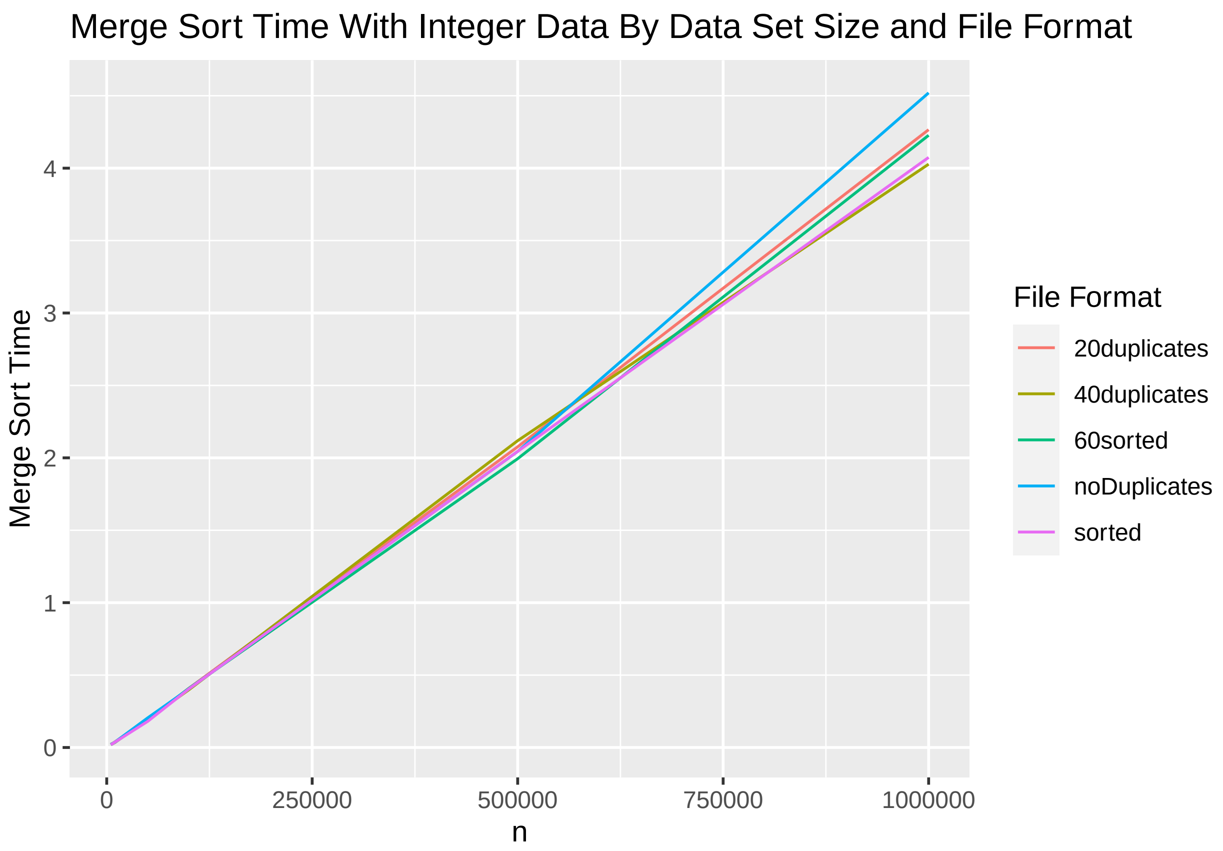


```
ggplot(mergeTimes, aes(x = nlogn, y = merge_time, color = var_type)) +
  labs(title = "Merge Sort Regression Models By Data Type", x = "nlogn", y = "Merge Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(6, 4)) +
  stat_cor(aes(label=..rr.label..),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

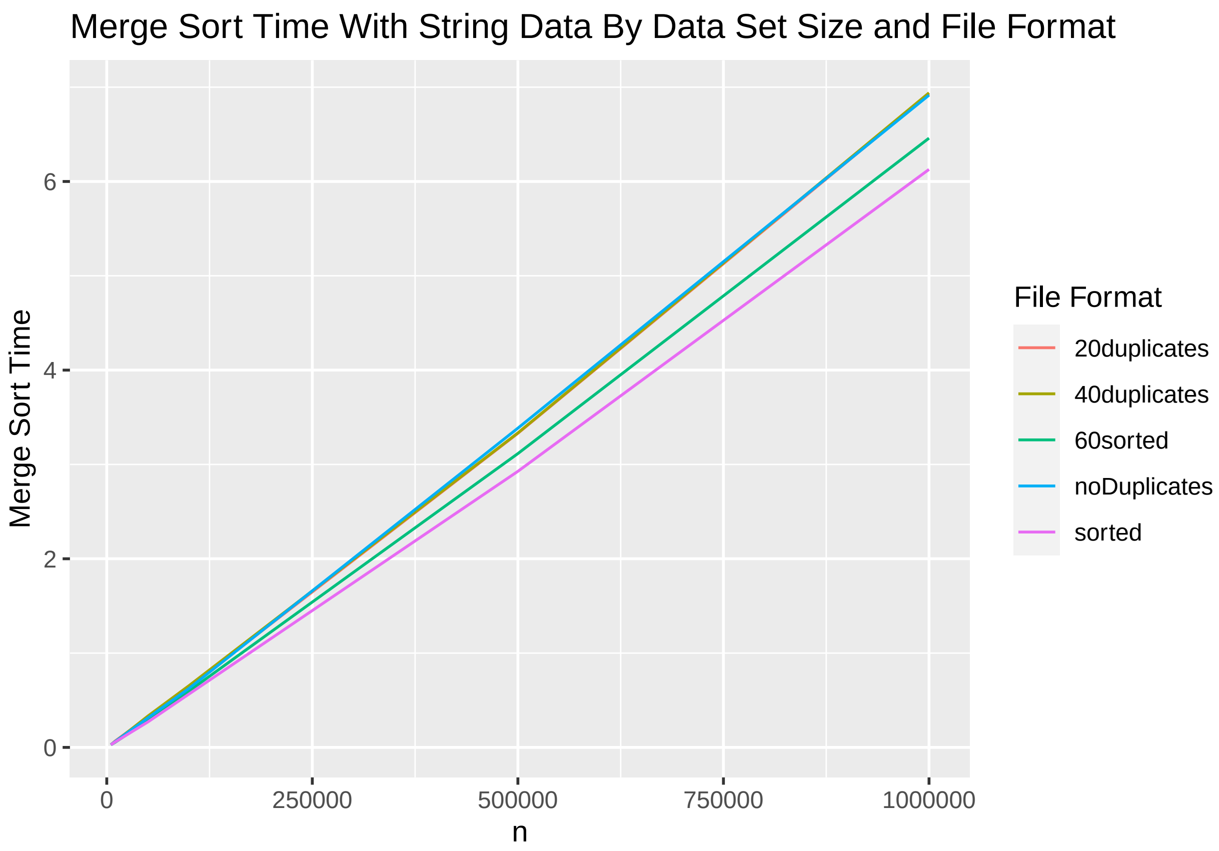
```
## `geom_smooth()` using formula 'y ~ x'
```



```r
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"))
```

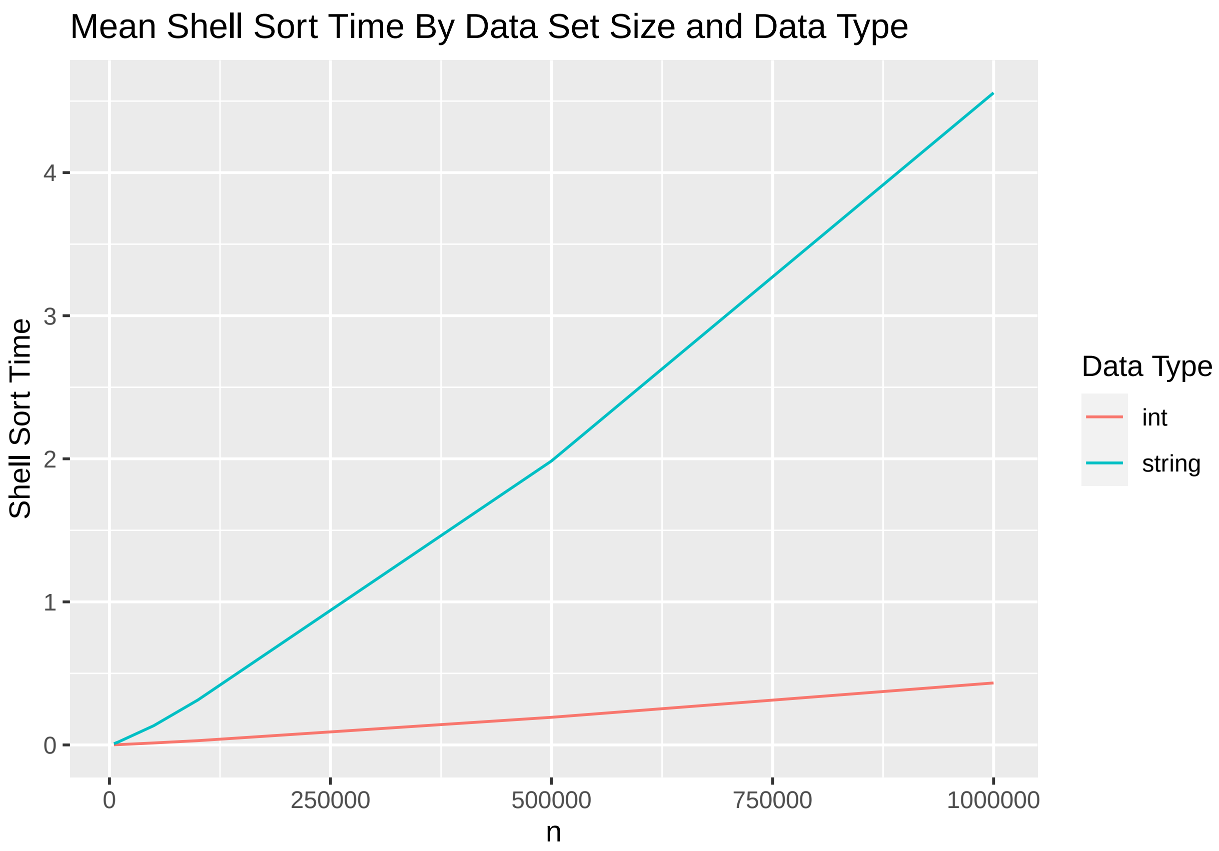


```
mergeStrings = subset(mergeTimes, var_type == "string")
ggplot(mergeStrings, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Merge
  guides(color = guide_legend(title = "File Format"))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



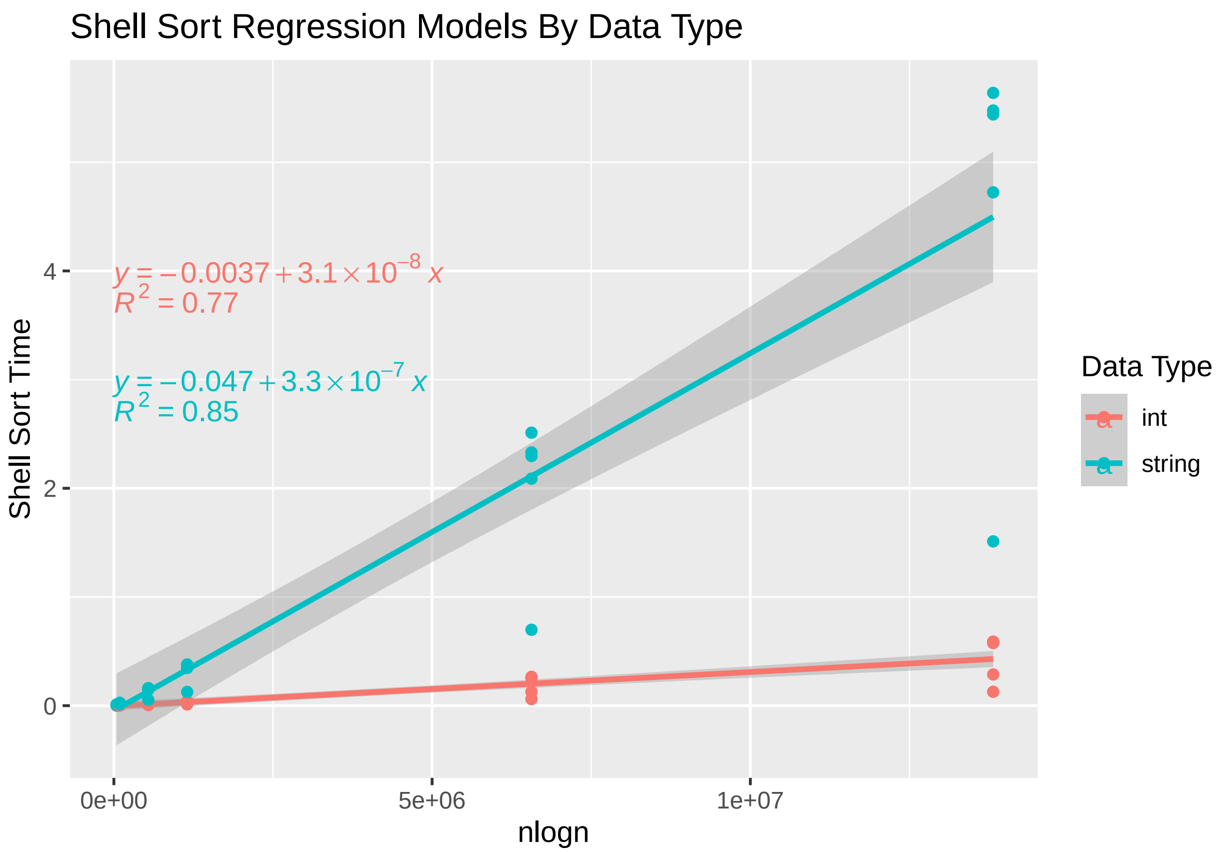
Shell Sort

```
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"))
```

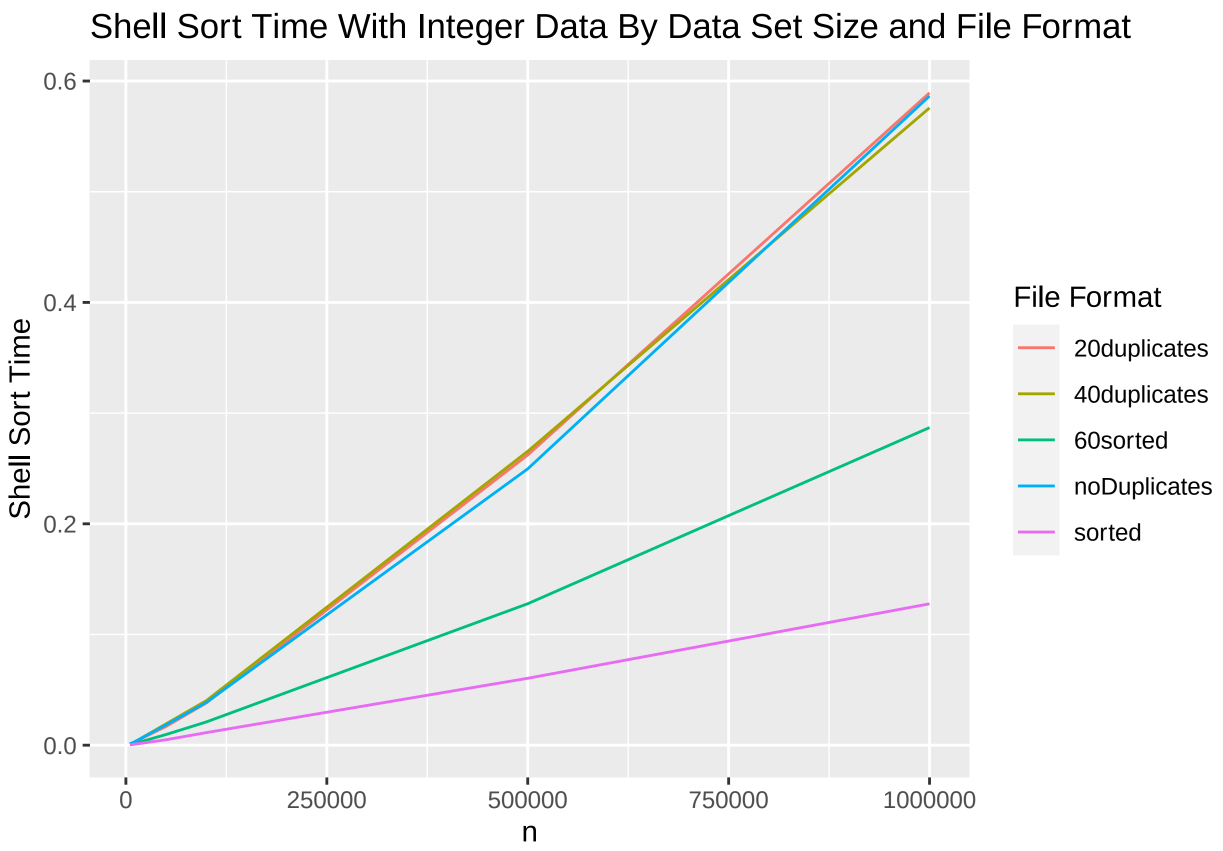


```
ggplot(shellTimes, aes(x = nlogn, y = shell_time, color = var_type)) +
  labs(title = "Shell Sort Regression Models By Data Type", x = "nlogn", y = "Shell Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

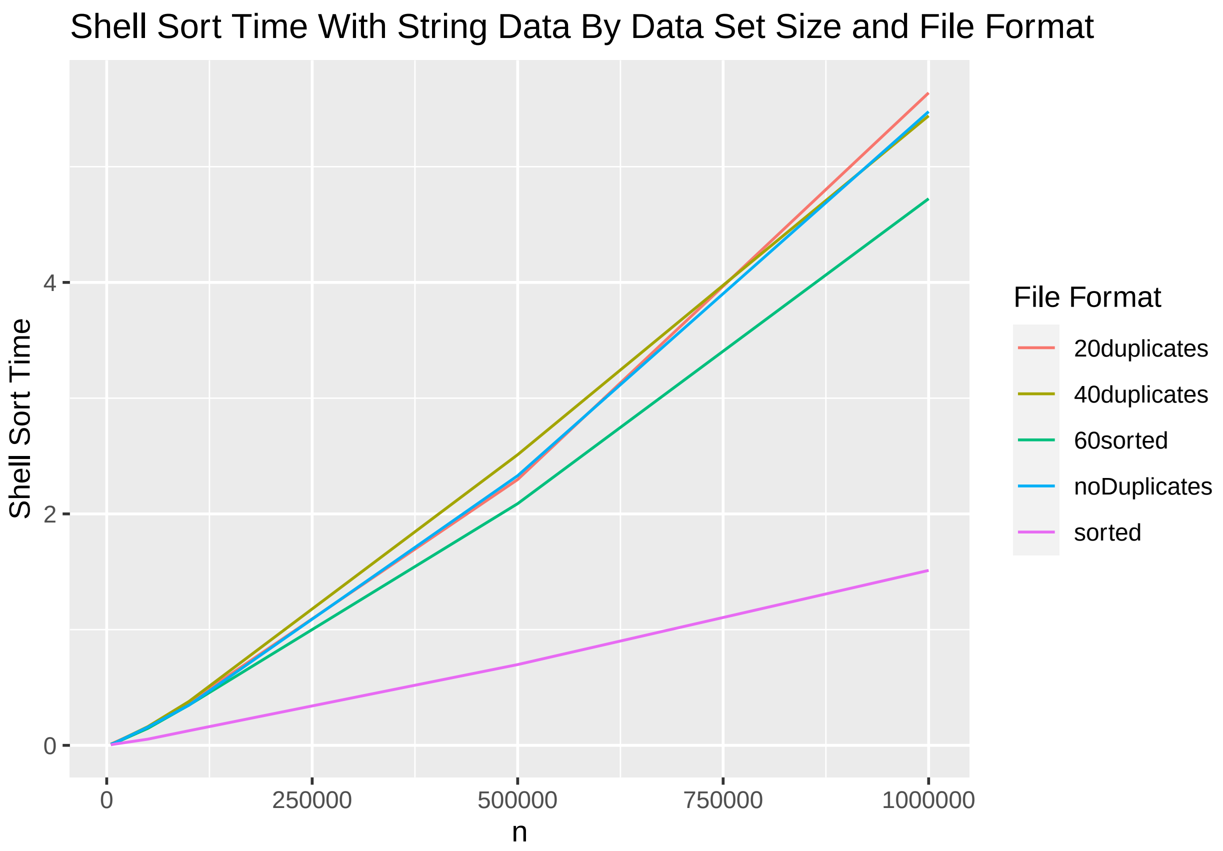
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

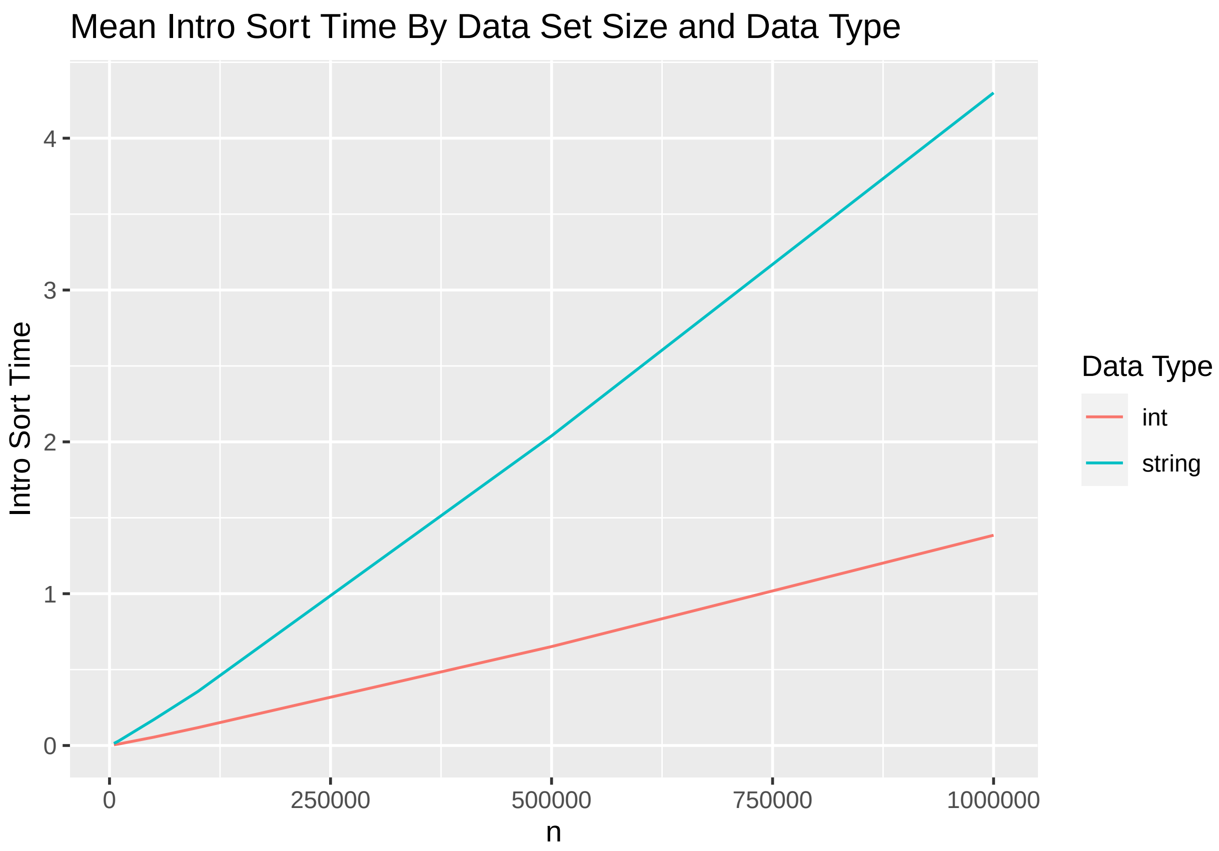


```
shellStrings = subset(shellTimes, var_type == "string")
ggplot(shellStrings, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Shell
  guides(color = guide_legend(title = "File Format"))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



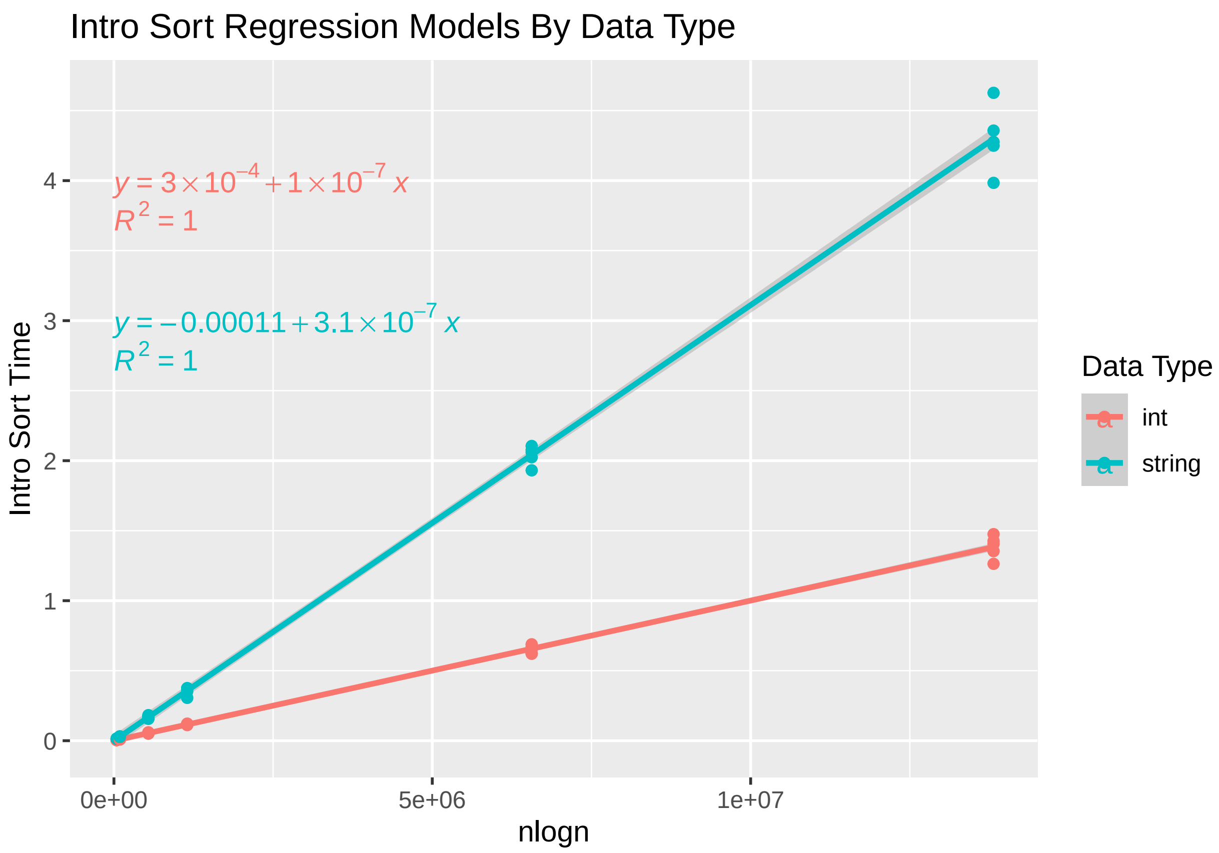
Intro Sort

```
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"))
```

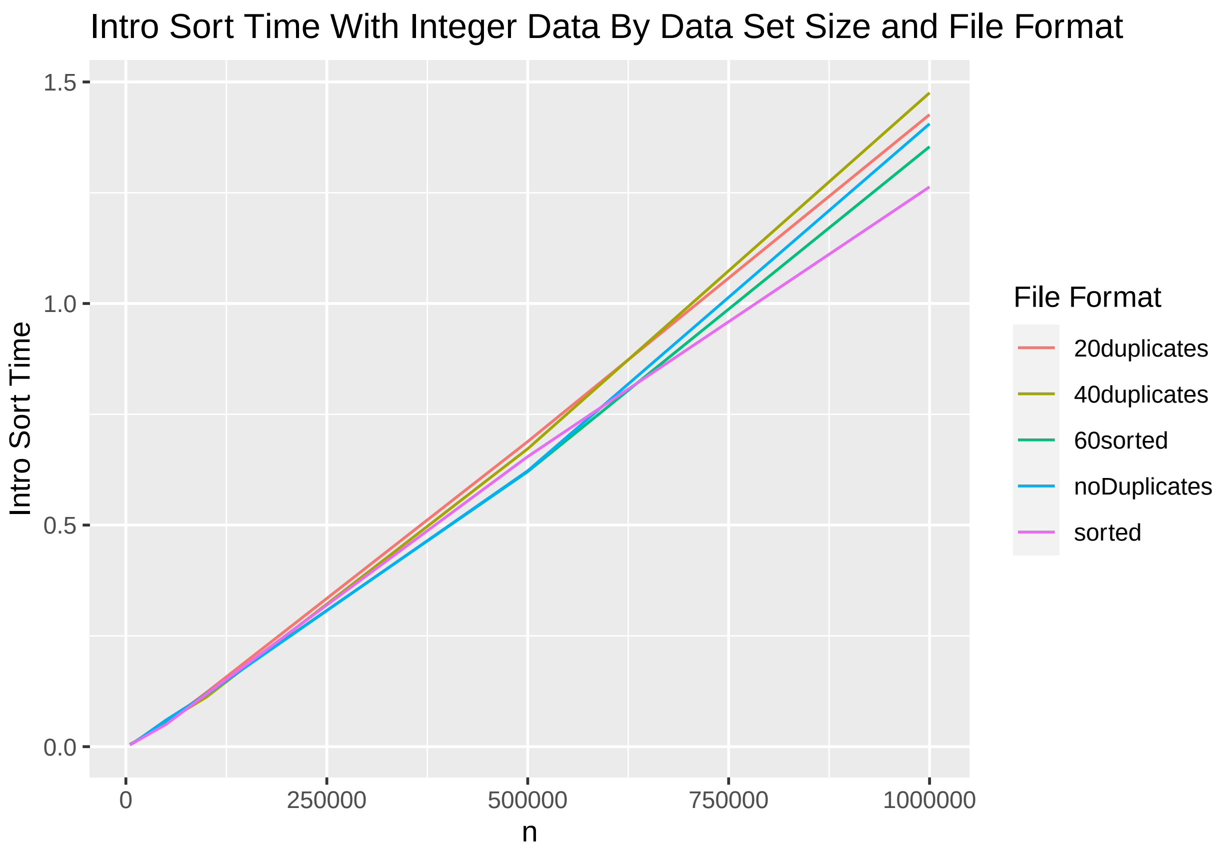


```
ggplot(introTimes, aes(x = nlogn, y = intro_time, color = var_type)) +
  labs(title = "Intro Sort Regression Models By Data Type", x = "nlogn", y = "Intro Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

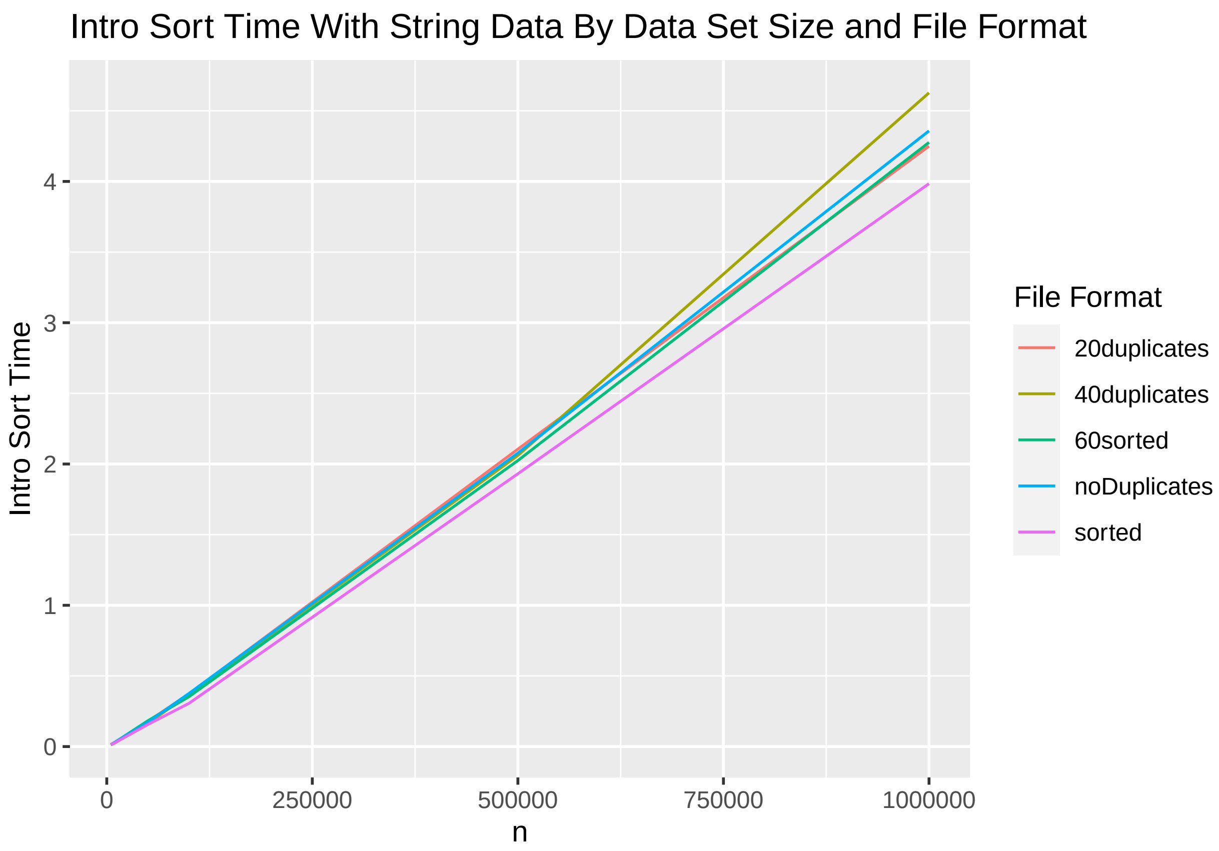
```
## `geom_smooth()` using formula 'y ~ x'
```



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

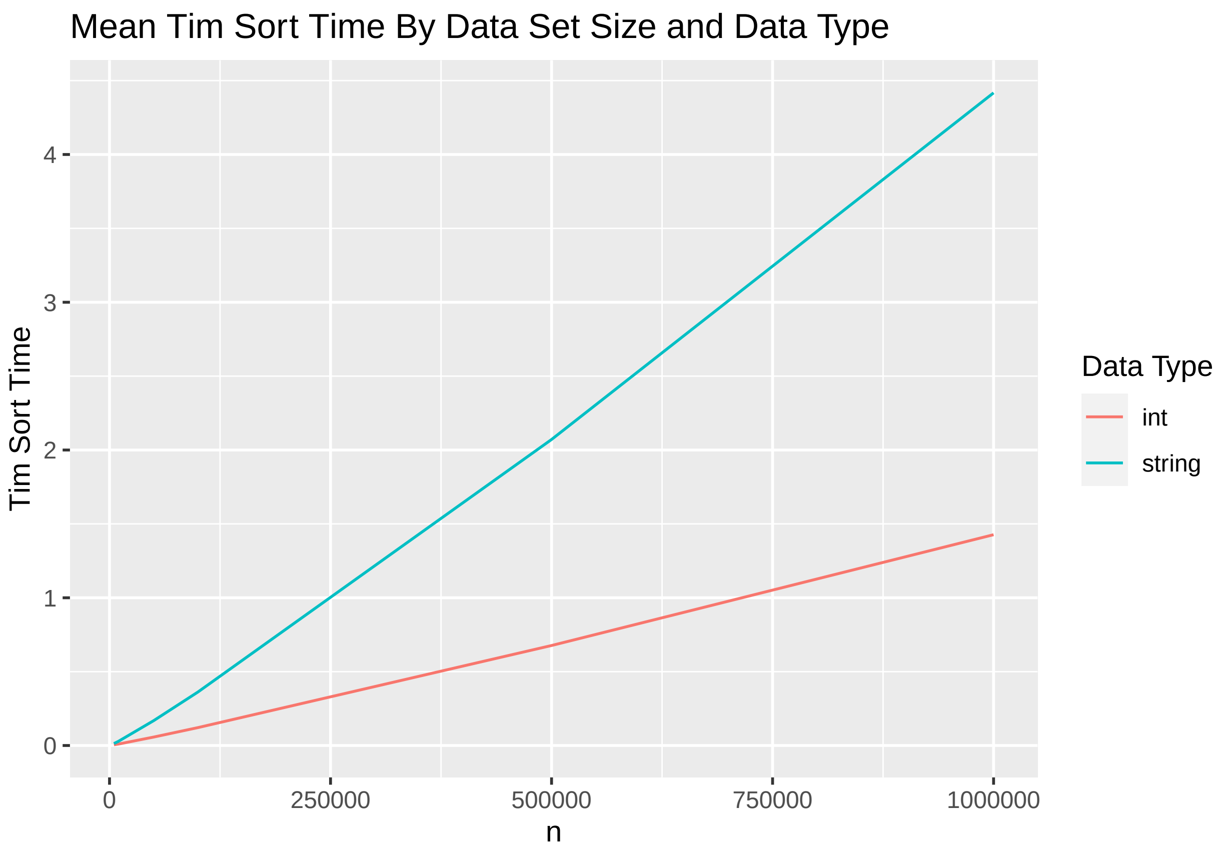


```
introStrings = subset(introTimes, var_type == "string")
ggplot(introStrings, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))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



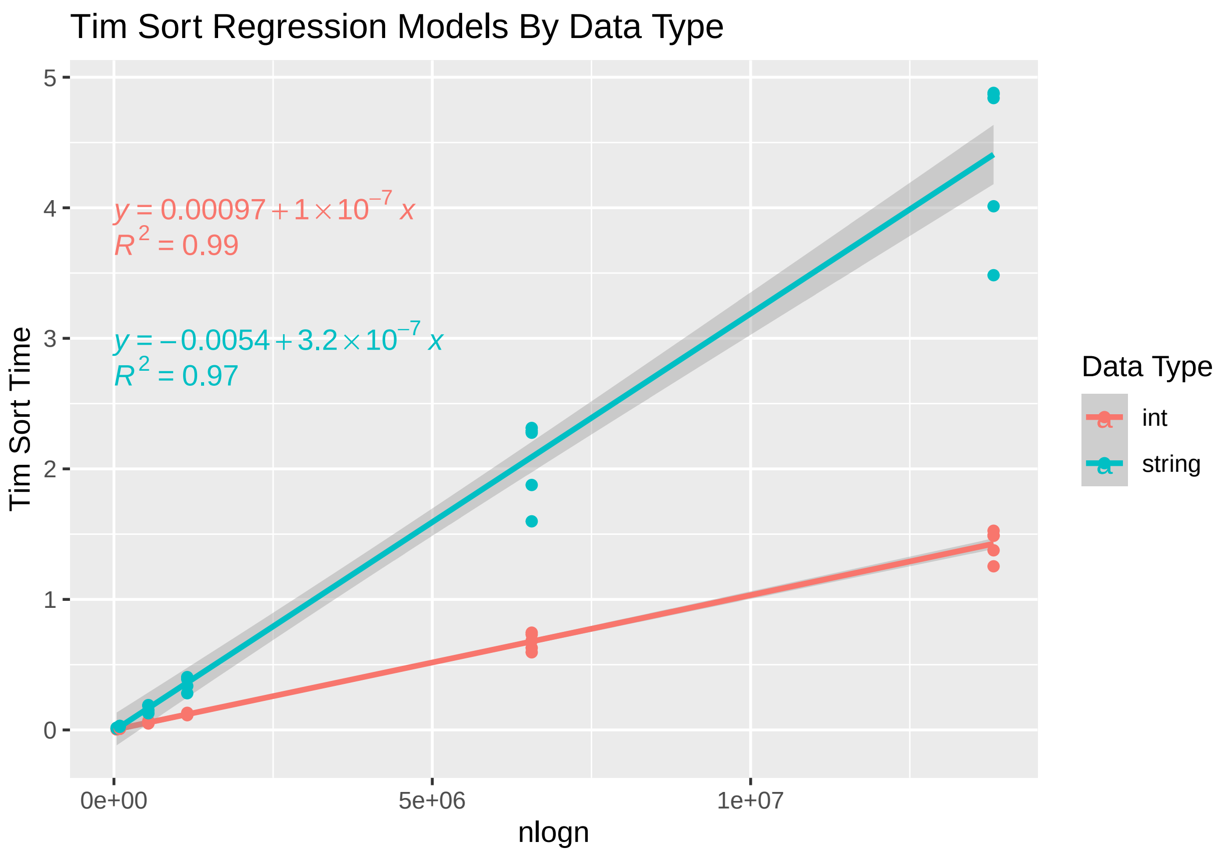
Tim Sort

```
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"))
```

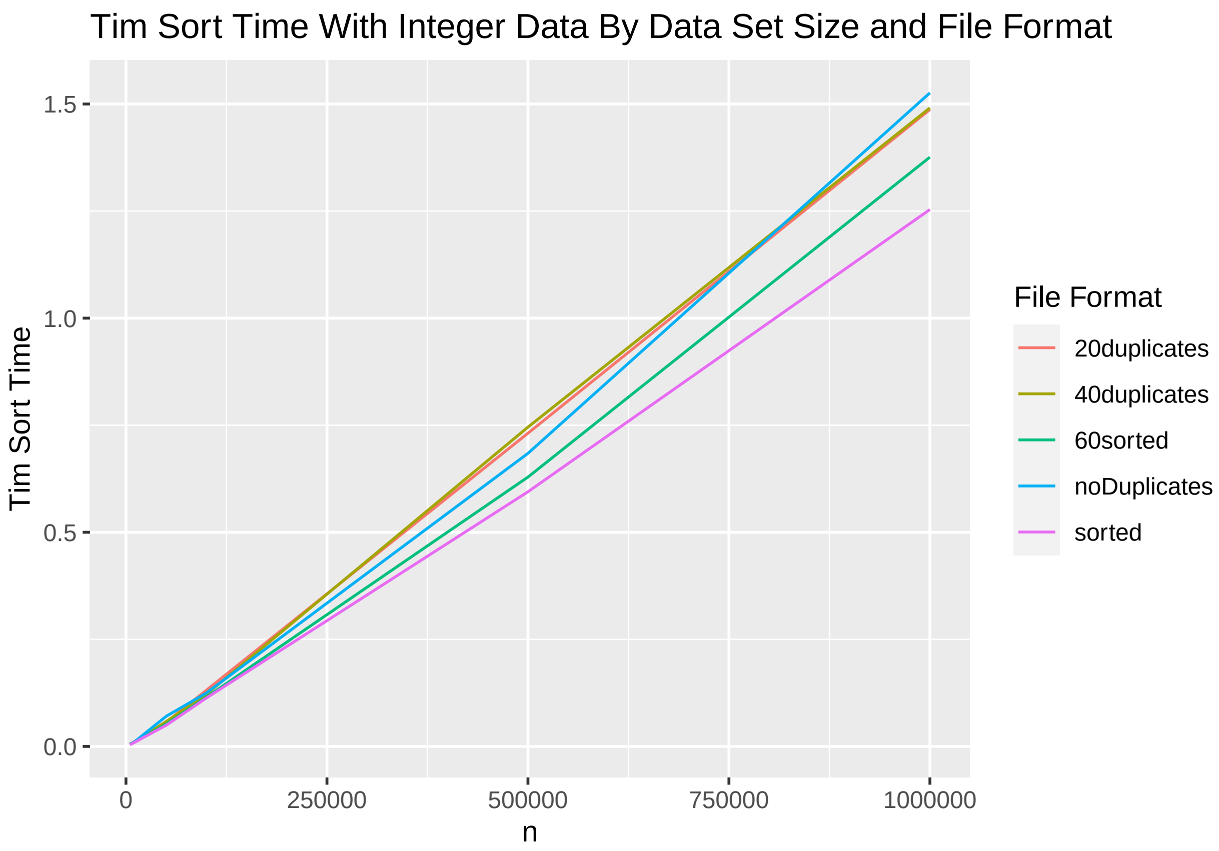


```r
ggplot(timTimes, aes(x = nlogn, y = tim_time, color = var_type)) +
  labs(title = "Tim Sort Regression Models By Data Type", x = "nlogn", y = "Tim Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

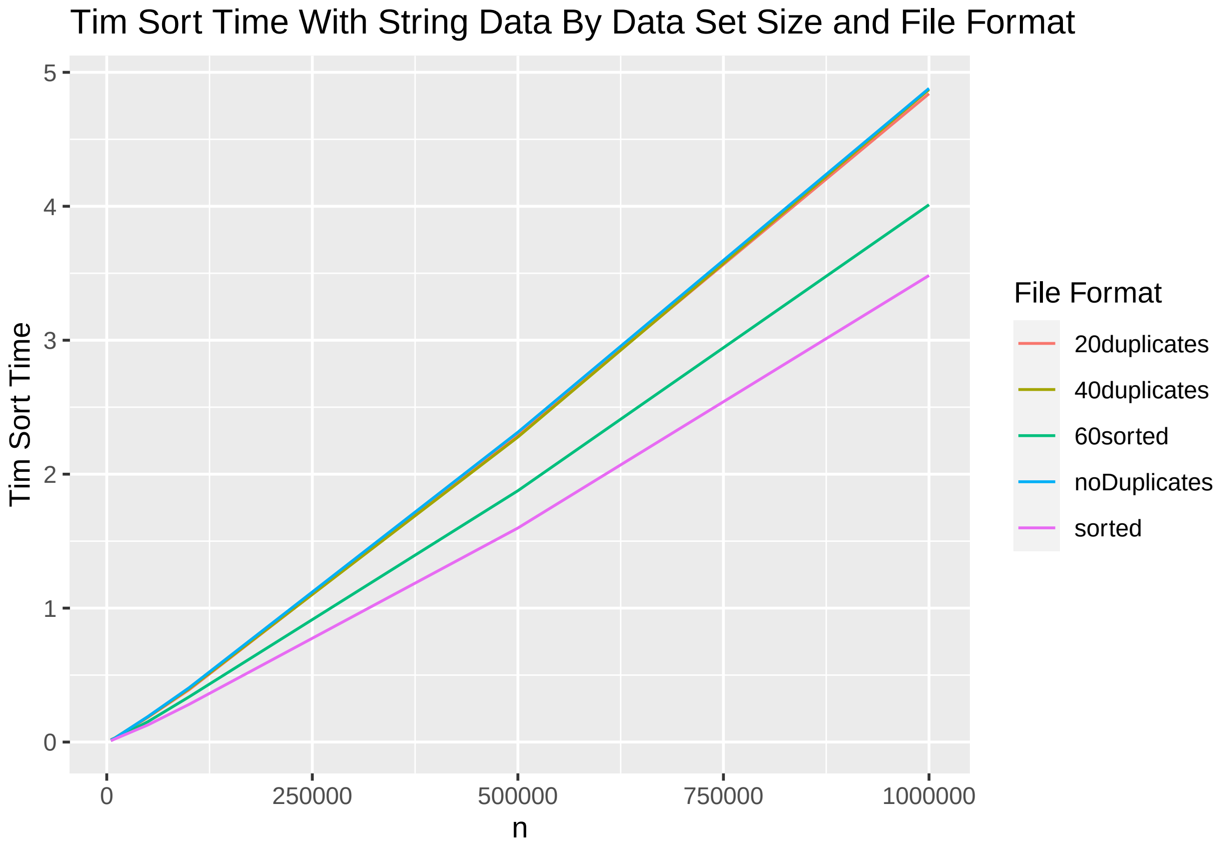
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```


Tim Sort Time With String Data By Data Set Size and File Format
Tim Sort Time
0
1
2
3
4
5
n
0
250000
500000
750000
1000000
File Format
20duplicates
40duplicates
60sorted
noDuplicates
sorted