Program 2 Graph Analysis

Ryan Schaefer and Wes Anderson

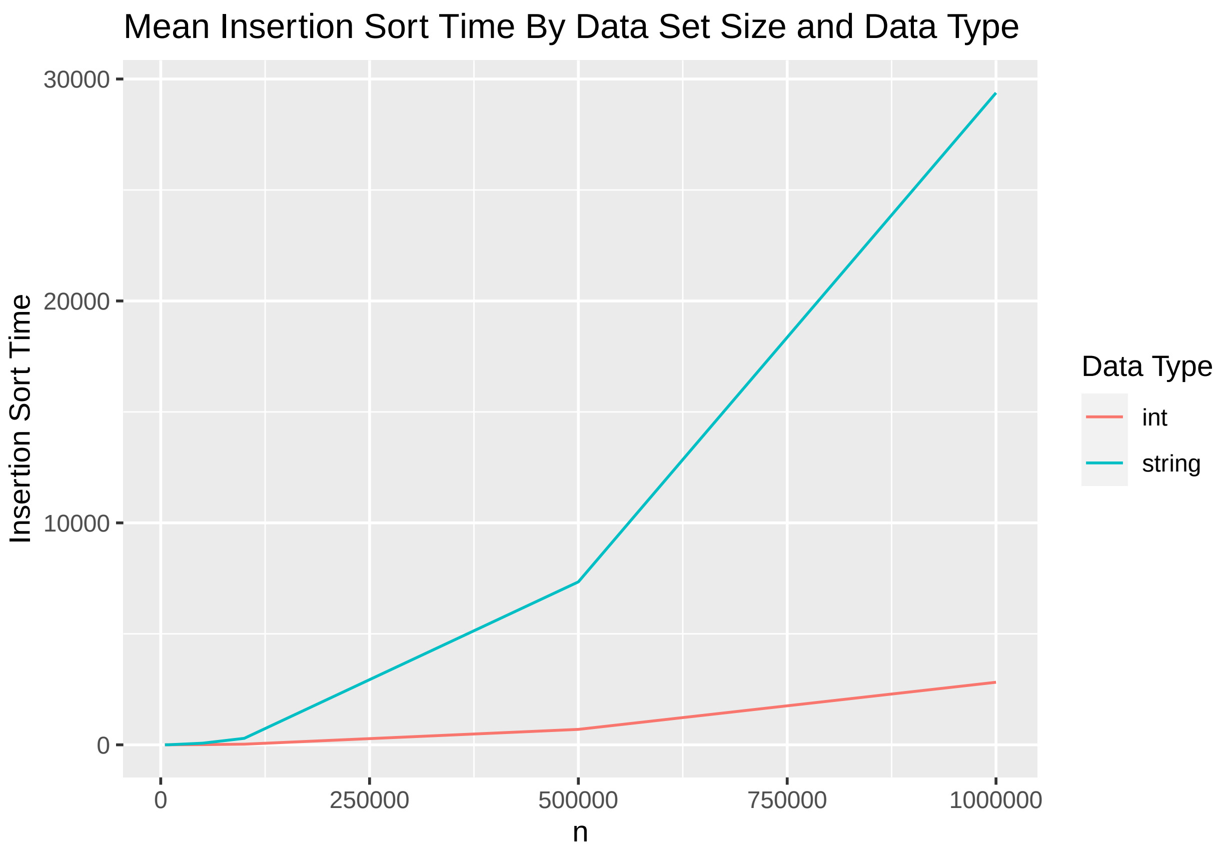
Create Dataset

```
library(ggplot2)
library(ggpubr)
data = read.csv("RyanMean.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    1.06218e+03 0.247838000  3.20655000
## 2       int 1000000  40duplicates    4.42405e+03 0.452143667  6.37824333
## 3       int  100000  40duplicates    4.40678e+01 0.039098933  0.61661700
## 4       int   10000  40duplicates    4.52373e-01 0.003447393  0.06398197
## 5       int   50000        sorted    1.07108e-03 0.017097167  0.31871033
## 6       int   50000  20duplicates    1.11540e+01 0.018411900  0.30000133
## 7       int    5000  noDuplicates    1.13868e-01 0.001630840  0.02721580
## 8       int  500000        sorted    1.00746e-02 0.145728000  3.36570000
## 9       int  500000      60sorted    1.78701e+02 0.193182333  3.49397333
## 10      int   10000      60sorted    7.53939e-02 0.003171413  0.06565017
## 11      int 1000000  noDuplicates    4.46894e+03 0.483649000  6.95366000
## 12      int 1000000  20duplicates    4.48015e+03 0.496550667  6.81604000
## 13      int   50000  noDuplicates    1.12508e+01 0.018315600  0.29182600
## 14      int    5000      60sorted    1.80125e-02 0.001332933  0.02587437
## 15      int    5000        sorted    9.82800e-05 0.001090583  0.02536007
## 16      int  100000  20duplicates    4.44996e+01 0.037988667  0.59879333
## 17      int   50000      60sorted    1.79186e+00 0.015144600  0.27707533
## 18      int   10000  noDuplicates    4.51946e-01 0.003456193  0.05796843
## 19      int  500000  20duplicates    1.11640e+03 0.214763667  2.99475667
## 20      int  500000  40duplicates    1.11854e+03 0.216381333  2.94043000
## 21      int 1000000        sorted    2.21053e-02 0.323354667  5.99160333
## 22      int    5000  20duplicates    1.11112e-01 0.001568040  0.02586533
## 23      int  100000  noDuplicates    4.45871e+01 0.037821067  0.58117633
## 24      int   50000  40duplicates    1.11186e+01 0.021332867  0.28594300
## 25      int   10000  20duplicates    4.48142e-01 0.003242267  0.05283480
## 26      int  100000        sorted    1.96880e-03 0.027465200  0.56341900
## 27      int  100000      60sorted    7.15581e+00 0.031533033  0.56025567
## 28      int   10000        sorted    2.00159e-04 0.002552103  0.05084763
## 29      int    5000  40duplicates    1.11897e-01 0.001578410  0.02579810
## 30      int 1000000      60sorted    7.15488e+02 0.389023000  6.32902667
## 31   string   50000        sorted    8.79662e-03 0.124609333  0.45050333
## 32   string  500000  20duplicates    1.02101e+04 1.603446667  5.81480000
## 33   string   50000  20duplicates    1.01569e+02 0.124722333  0.53288100
## 34   string   10000 40duplicates    4.08111e+00 0.020698367  0.09355417
## 35   string   10000     60sorted    2.59330e+00 0.020809400  0.08908853
## 36   string  100000       sorted    1.85365e-02 0.249897333  0.93209567
## 37   string    5000 40duplicates    9.99780e-01 0.009321353  0.04433850
## 38   string  500000     60sorted    6.55264e+03 1.510433333  5.06751333
## 39   string   50000 noDuplicates    9.61621e+01 0.123088333  0.50066967
## 40   string  500000 40duplicates    9.88785e+03 1.603070000  5.53600333
## 41   string    5000 20duplicates    9.81218e-01 0.009668423  0.04428087
## 42   string  100000 noDuplicates    3.95303e+02 0.268085000  1.05147667
## 43   string    5000 noDuplicates    9.94519e-01 0.009564517  0.04481913
## 44   string  100000     60sorted    2.53332e+02 0.254192667  0.96740267
## 45   string 1000000 20duplicates    4.01252e+04 3.308846667 11.54003333
## 46   string   10000 noDuplicates    4.01830e+00 0.022386400  0.09470773
## 47   string 1000000 noDuplicates    4.04289e+04 3.233970000 11.37780000
## 48   string 1000000       sorted    1.84133e-01 3.145206667  9.83583333
## 49   string  500000 noDuplicates    1.00548e+04 1.558813333  5.49725000
## 50   string  100000 40duplicates    4.00866e+02 0.272295333  1.05728000
## 51   string    5000     60sorted    6.27955e-01 0.009499383  0.04299257
## 52   string 1000000     60sorted    2.57302e+04 3.172463333 10.56360000
## 53   string    5000       sorted    8.75481e-04 0.009027940  0.04032363
## 54   string  100000 20duplicates    4.08117e+02 0.271641333  1.05253333
## 55   string   10000 20duplicates    4.05438e+00 0.019917967  0.09235450
## 56   string   10000       sorted    1.84725e-03 0.020702900  0.08134747
## 57   string  500000       sorted    9.52993e-02 1.478916667  4.89594667
## 58   string   50000 40duplicates    1.01896e+02 0.128892667  0.50431767
## 59   string   50000     60sorted    6.58944e+01 0.127282667  0.46627633
## 60   string 1000000 40duplicates    4.05995e+04 3.226340000 11.44600000
##      shell_time  intro_time    tim_time      n2       nlogn
## 1  0.455514000 1.050055000 1.199856667 2.5e+11  6561181.69
## 2  0.944439333 2.381526667 2.517253333 1.0e+12 13815510.56
## 3  0.072308733 0.213413333 0.241423667 1.0e+10  1151292.55
## 4  0.004543083 0.017783067 0.020423133 1.0e+08    92103.40
## 5  0.008760537 0.088832000 0.085176567 2.5e+09   540988.91
## 6  0.029030000 0.095837333 0.106089867 2.5e+09   540988.91
## 7  0.001895043 0.007244113 0.008006727 2.5e+07    42585.97
## 8  0.124369000 1.243006667 1.232230000 2.5e+11  6561181.69
## 9  0.236833000 1.218836667 1.257150000 2.5e+11  6561181.69
## 10 0.002847693 0.019267733 0.019103900 1.0e+08    92103.40
## 11 1.049099667 2.493913333 2.737380000 1.0e+12 13815510.56
## 12 1.005424000 2.412953333 2.553573333 1.0e+12 13815510.56
## 13 0.029825700 0.090156800 0.098048267 2.5e+09   540988.91
## 14 0.001164230 0.006761683 0.007005850 2.5e+07    42585.97
## 15 0.000683382 0.006379383 0.006814847 2.5e+07    42585.97
## 16 0.065054567 0.190086667 0.219187667 1.0e+10  1151292.55
## 17 0.016292033 0.086419367 0.088389300 2.5e+09   540988.91
## 18 0.004730367 0.017628933 0.020016033 1.0e+08    92103.40
## 19 0.437221667 1.056440000 1.163666667 2.5e+11  6561181.69
## 20 0.422426333 1.059106667 1.138090000 2.5e+11  6561181.69
## 21 0.235729667 2.162610000 2.117993333 1.0e+12 13815510.56
## 22 0.001834977 0.006966713 0.007716657 2.5e+07    42585.97
## 23 0.067271267 0.191841667 0.205473333 1.0e+10  1151292.55
## 24 0.028864033 0.090209400 0.095657500 2.5e+09   540988.91
## 25 0.004259540 0.014528967 0.017369667 1.0e+08    92103.40
## 26 0.019828100 0.178087333 0.180310000 1.0e+10  1151292.55
## 27 0.036369533 0.192777667 0.186424667 1.0e+10  1151292.55
## 28 0.001489380 0.015154100 0.013952267 1.0e+08    92103.40
## 29 0.002021813 0.007955273 0.009012693 2.5e+07    42585.97
## 30 0.480602000 2.260596667 2.270166667 1.0e+12 13815510.56
## 31 0.100705033 0.248759333 0.228166333 2.5e+09   540988.91
## 32 4.027273333 3.344520000 4.118650000 2.5e+11  6561181.69
## 33 0.263552667 0.269618000 0.339872000 2.5e+09   540988.91
## 34 0.036852000 0.044239333 0.059170833 1.0e+08    92103.40
## 35 0.038245700 0.044978600 0.046503567 1.0e+08    92103.40
## 36 0.220999667 0.521138667 0.491032667 1.0e+10  1151292.55
## 37 0.015751633 0.019205167 0.027814200 2.5e+07    42585.97
## 38 3.453676667 3.076346667 3.249566667 2.5e+11  6561181.69
## 39 0.266975667 0.248294000 0.327950667 2.5e+09   540988.91
## 40 4.168180000 3.282103333 4.009226667 2.5e+11  6561181.69
## 41 0.015803533 0.021263100 0.025350433 2.5e+07    42585.97
## 42 0.590983667 0.549322000 0.709278000 1.0e+10  1151292.55
## 43 0.016193333 0.020278733 0.025569067 2.5e+07    42585.97
## 44 0.566019667 0.534172333 0.578343333 1.0e+10  1151292.55
## 45 9.463143333 6.838210000 8.472980000 1.0e+12 13815510.56
## 46 0.038911333 0.053003067 0.058801000 1.0e+08    92103.40
## 47 9.148363333 6.677910000 8.399910000 1.0e+12 13815510.56
## 48 2.509186667 6.380866667 5.859376667 1.0e+12 13815510.56
## 49 3.838886667 3.120976667 4.050583333 2.5e+11  6561181.69
## 50 0.604183000 0.546589333 0.707953667 1.0e+10  1151292.55
## 51 0.014272900 0.019154033 0.021220600 2.5e+07    42585.97
## 52 7.806923333 6.794706667 7.042206667 1.0e+12 13815510.56
## 53 0.007872937 0.018695600 0.017291900 2.5e+07    42585.97
## 54 0.626764667 0.601872667 0.720642667 1.0e+10  1151292.55
## 55 0.036663767 0.041434600 0.056526033 1.0e+08    92103.40
## 56 0.016622467 0.041451033 0.044311133 1.0e+08    92103.40
## 57 1.218546667 3.033643333 2.808856667 2.5e+11  6561181.69
## 58 0.265339000 0.260078000 0.330697333 2.5e+09   540988.91
## 59 0.248833667 0.251263667 0.272039000 2.5e+09   540988.91
## 60 8.987036667 6.857020000 8.594590000 1.0e+12 13815510.56
```

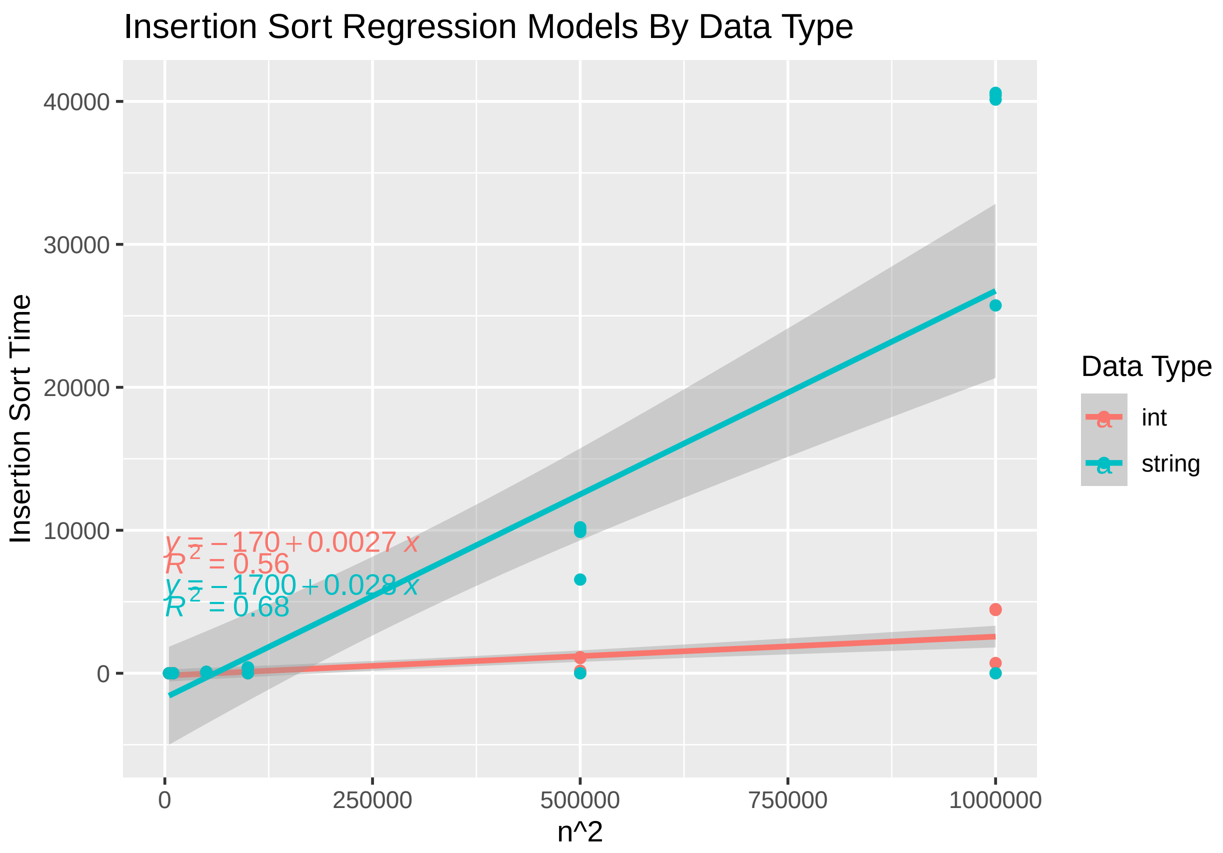
Insertion Sort

```
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 
  guides(color = guide_legend(title = "Data Type"))
```

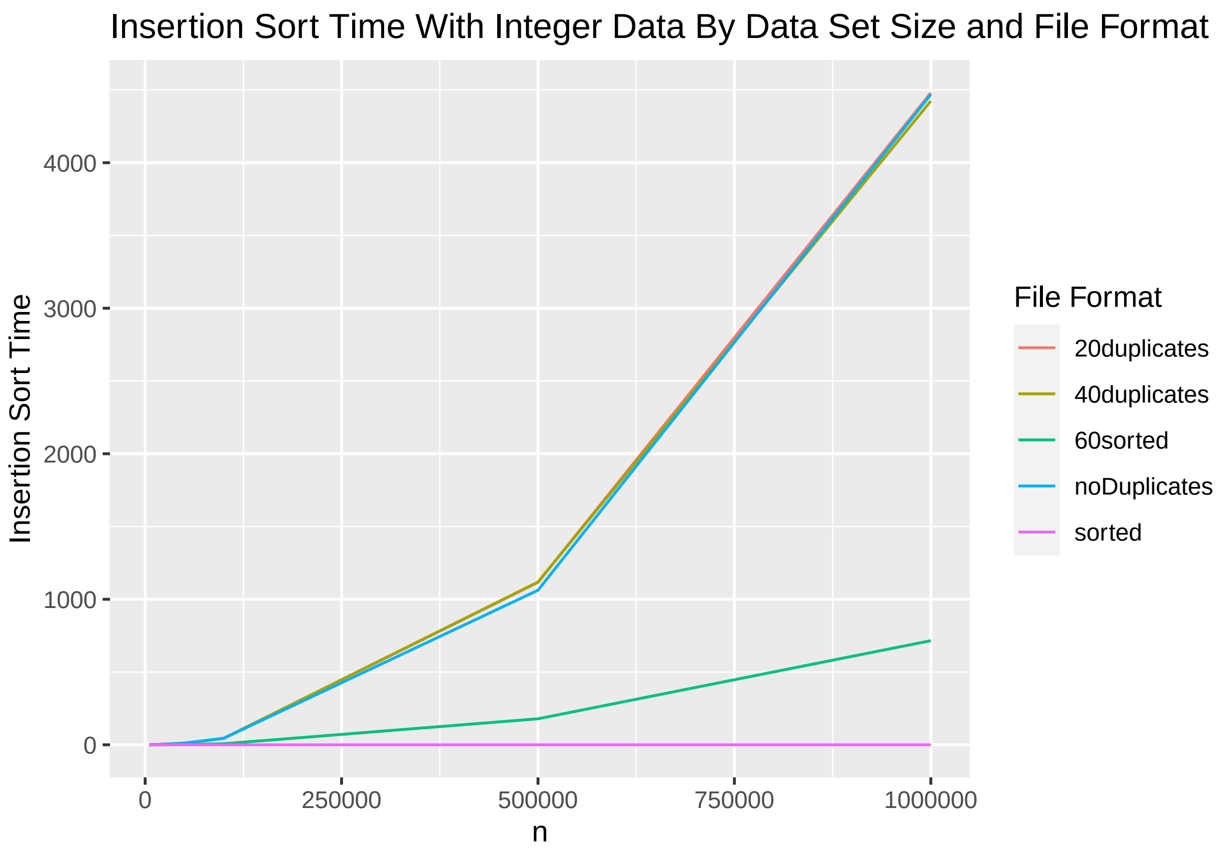


```
ggplot(insertionTimes, aes(x = size, y = insertion_time, color = var_type)) +
  labs(title = "Insertion Sort Regression Models By Data Type", x = "n^2", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(9000, 6000)) +
  stat_cor(aes(label=..rr.label..),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

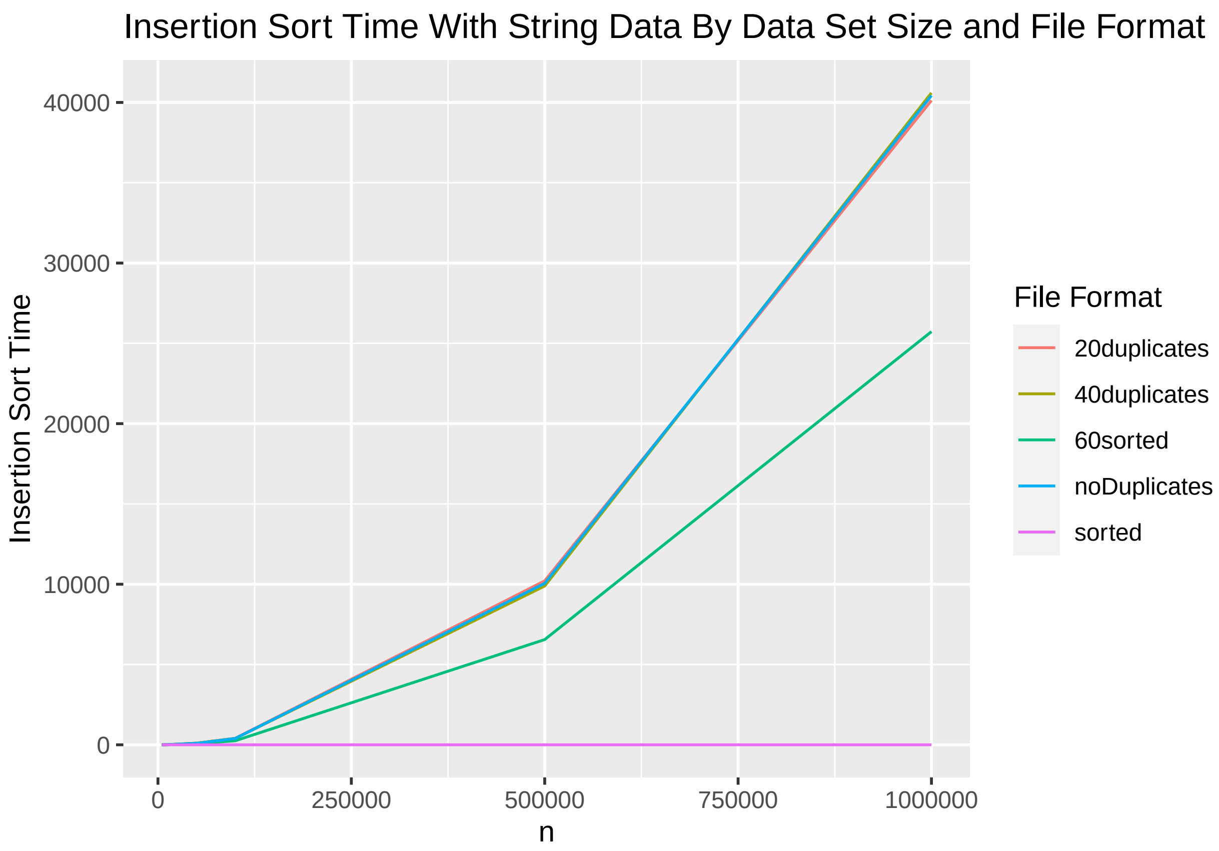
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

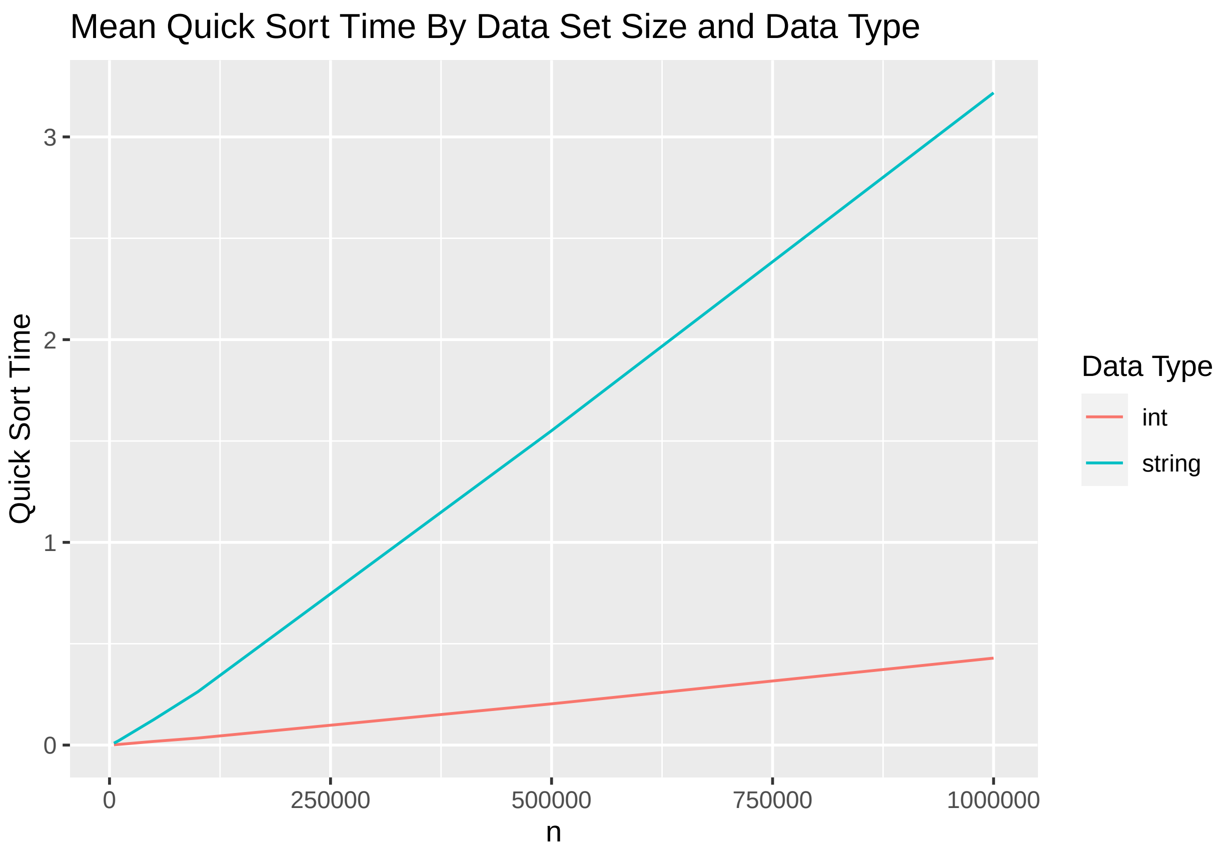


```
insertionStrings = subset(insertionTimes, var_type == "string")
ggplot(insertionStrings, aes(x = size, y = insertion_time, color = format)) +
  geom_line() +
  labs(title = "Insertion Sort Time With String Data By Data Set Size and File Format", x = "n", y = "In
  guides(color = guide_legend(title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



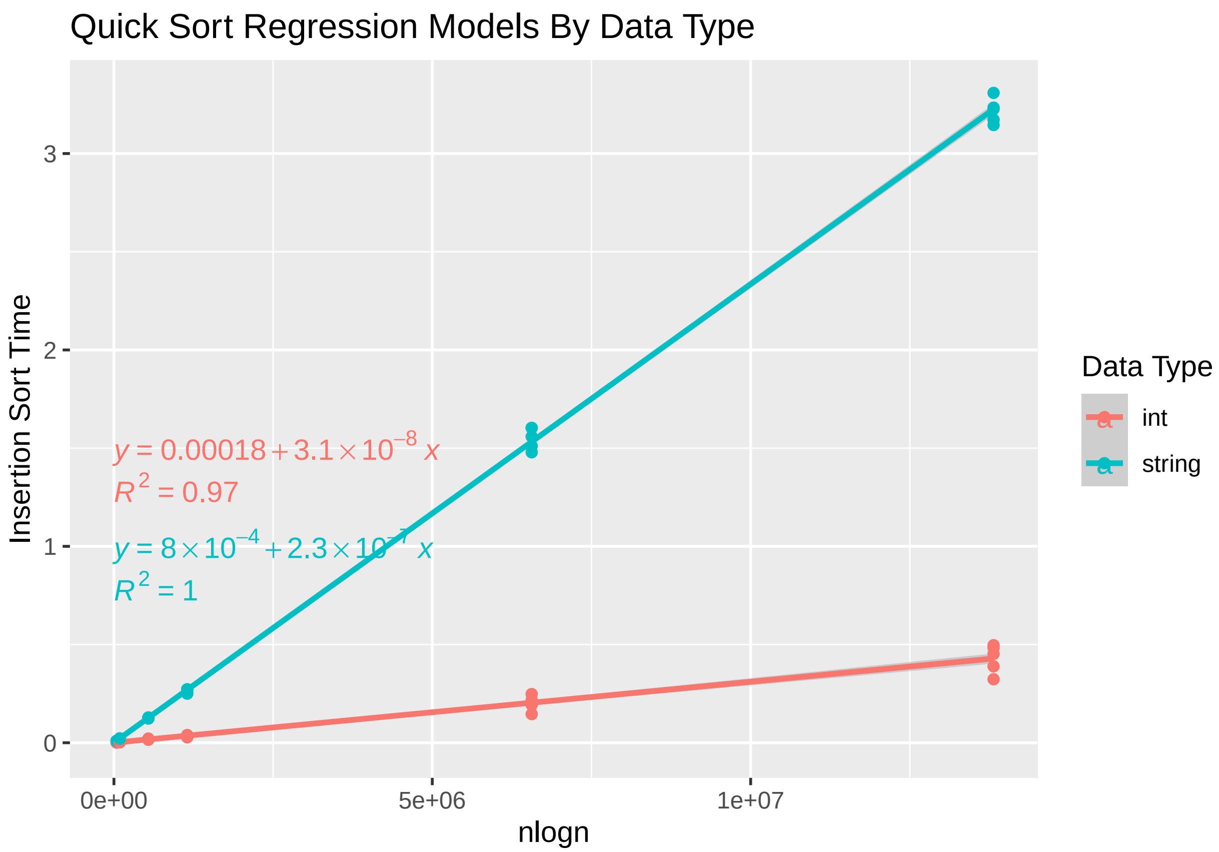
Quick Sort

```
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"))
```

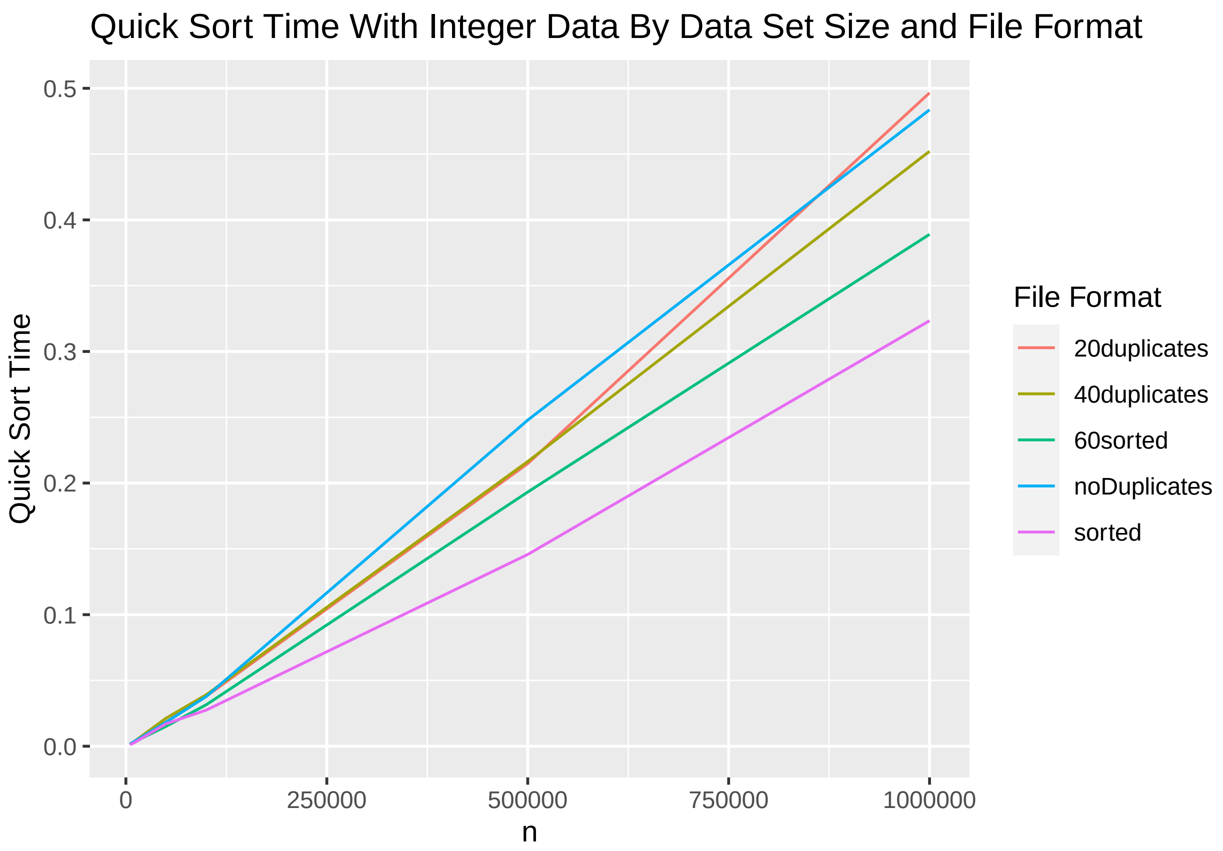


```r
ggplot(quickTimes, aes(x = nlogn, y = quick_time, color = var_type)) +
  labs(title = "Quick Sort Regression Models By Data Type", x = "nlogn", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(1.5, 1)) +
  stat_cor(aes(label=..rr.label..),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

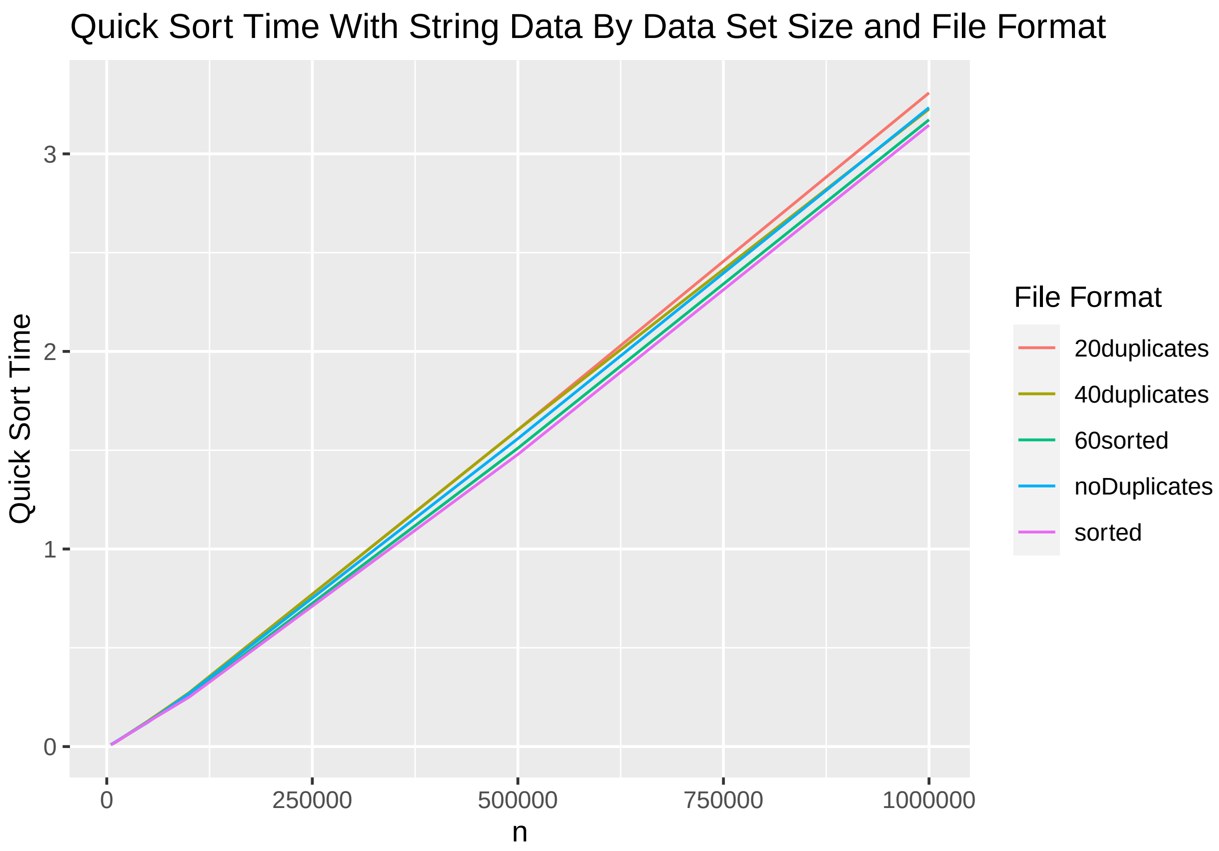
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

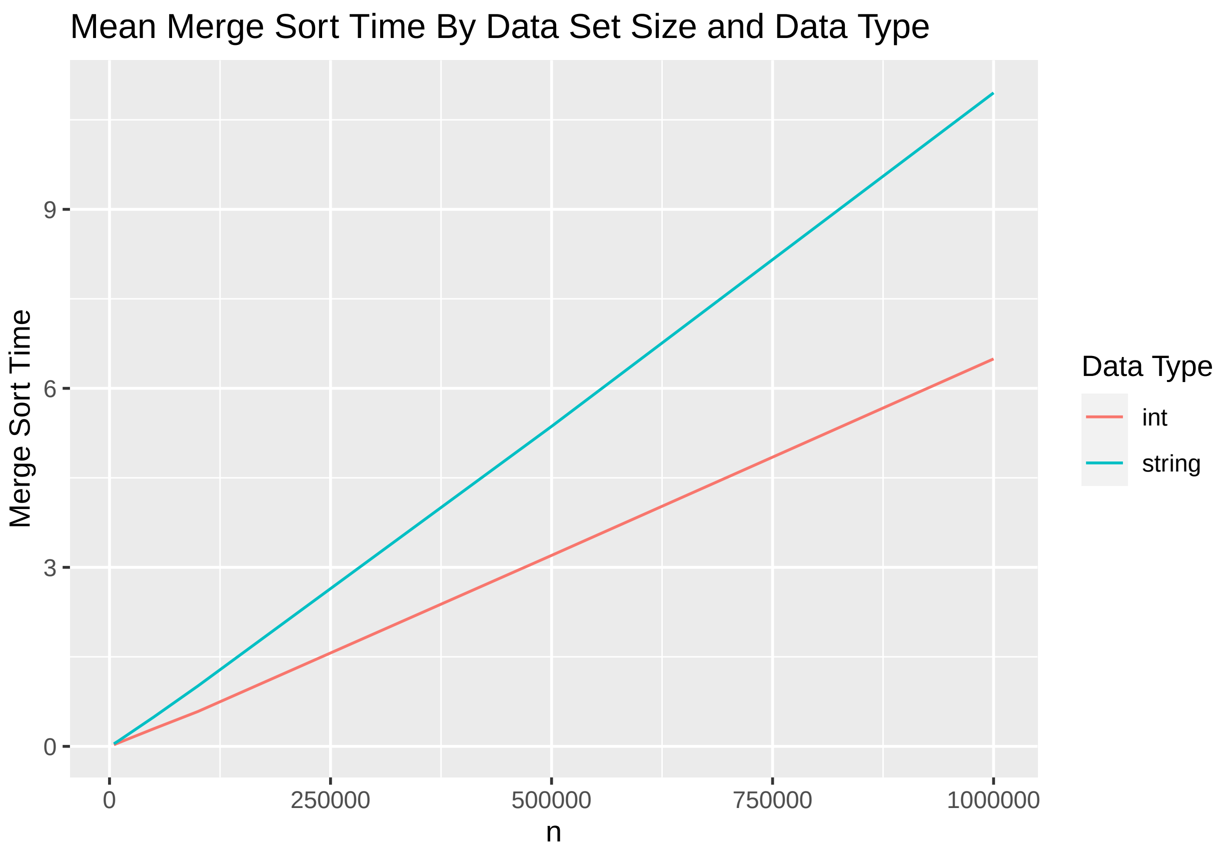


```
quickStrings = subset(quickTimes, var_type == "string")
ggplot(quickStrings, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Quick
  guides(color = guide_legend(title = "File Format"))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



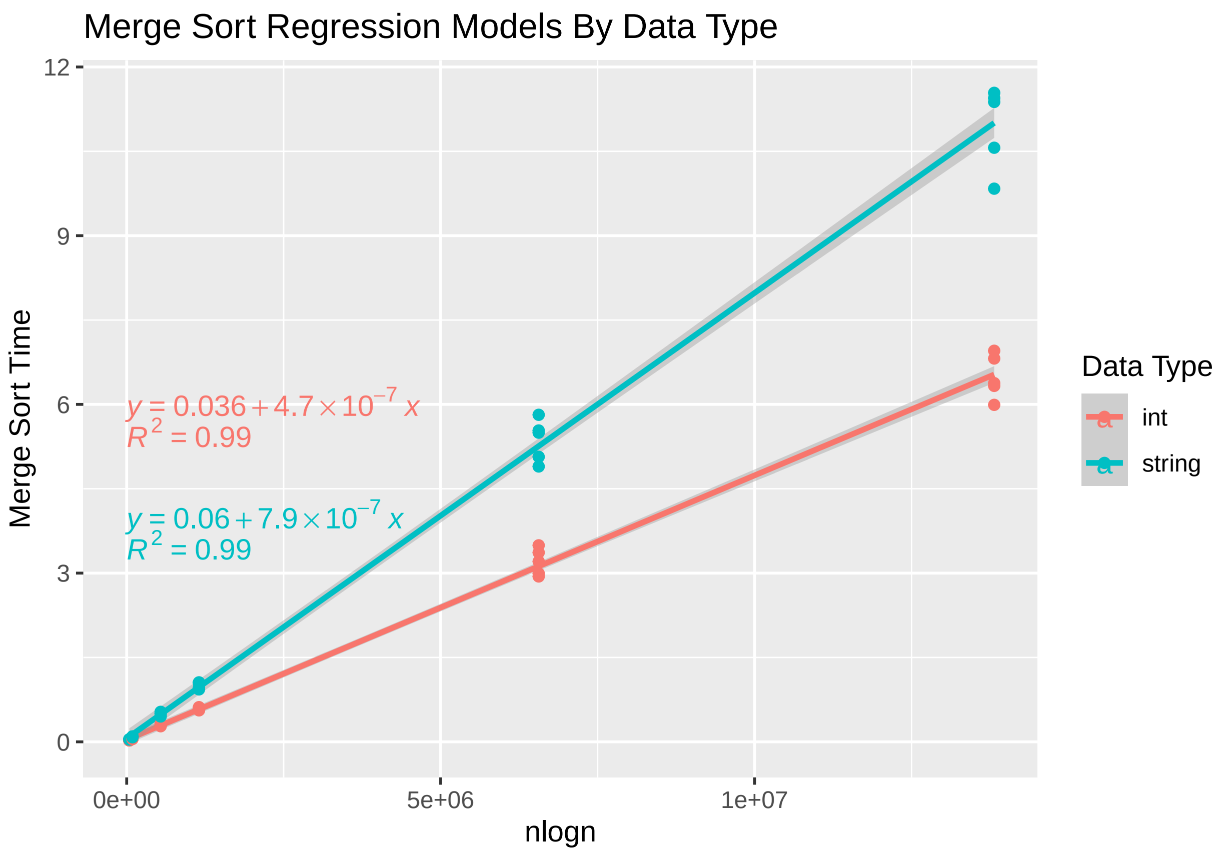
Merge Sort

```
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"))
```

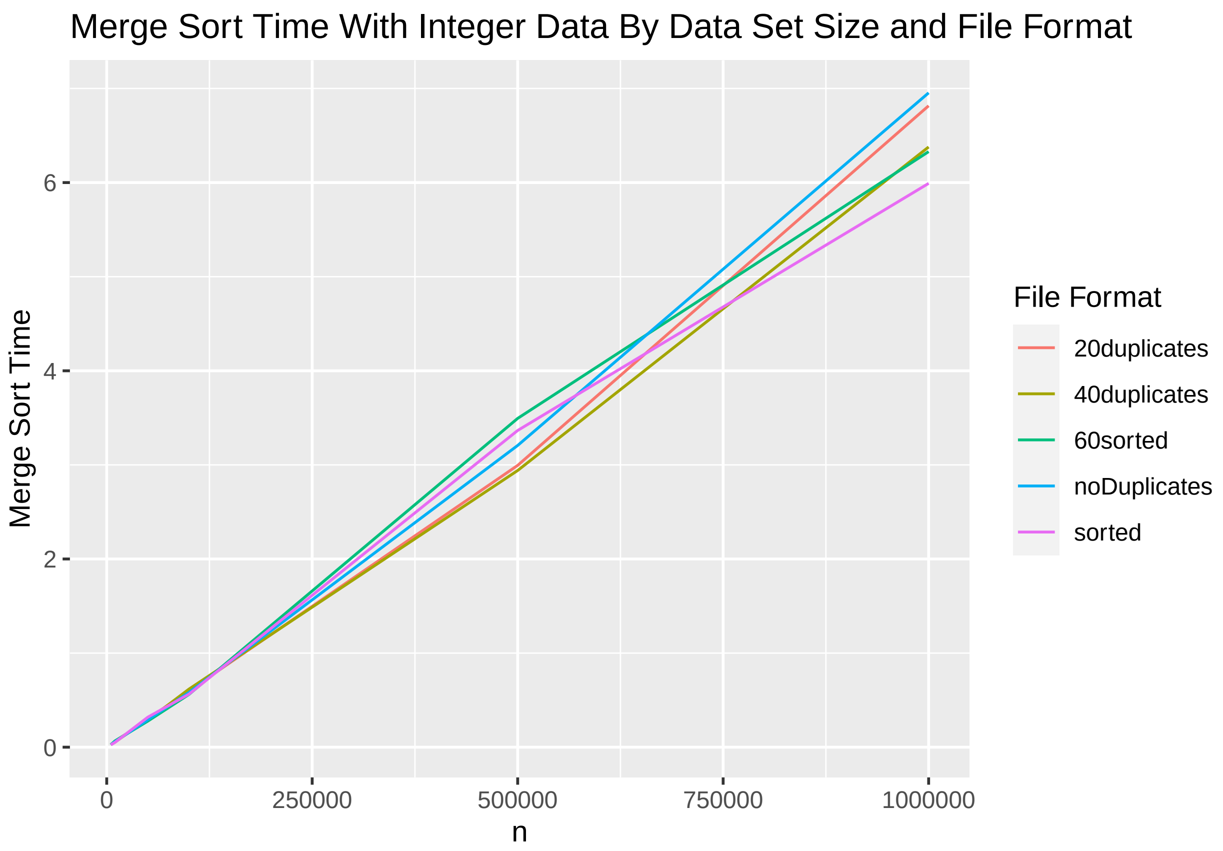


```
ggplot(mergeTimes, aes(x = nlogn, y = merge_time, color = var_type)) +
  labs(title = "Merge Sort Regression Models By Data Type", x = "nlogn", y = "Merge Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(6, 4)) +
  stat_cor(aes(label=..rr.label..),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

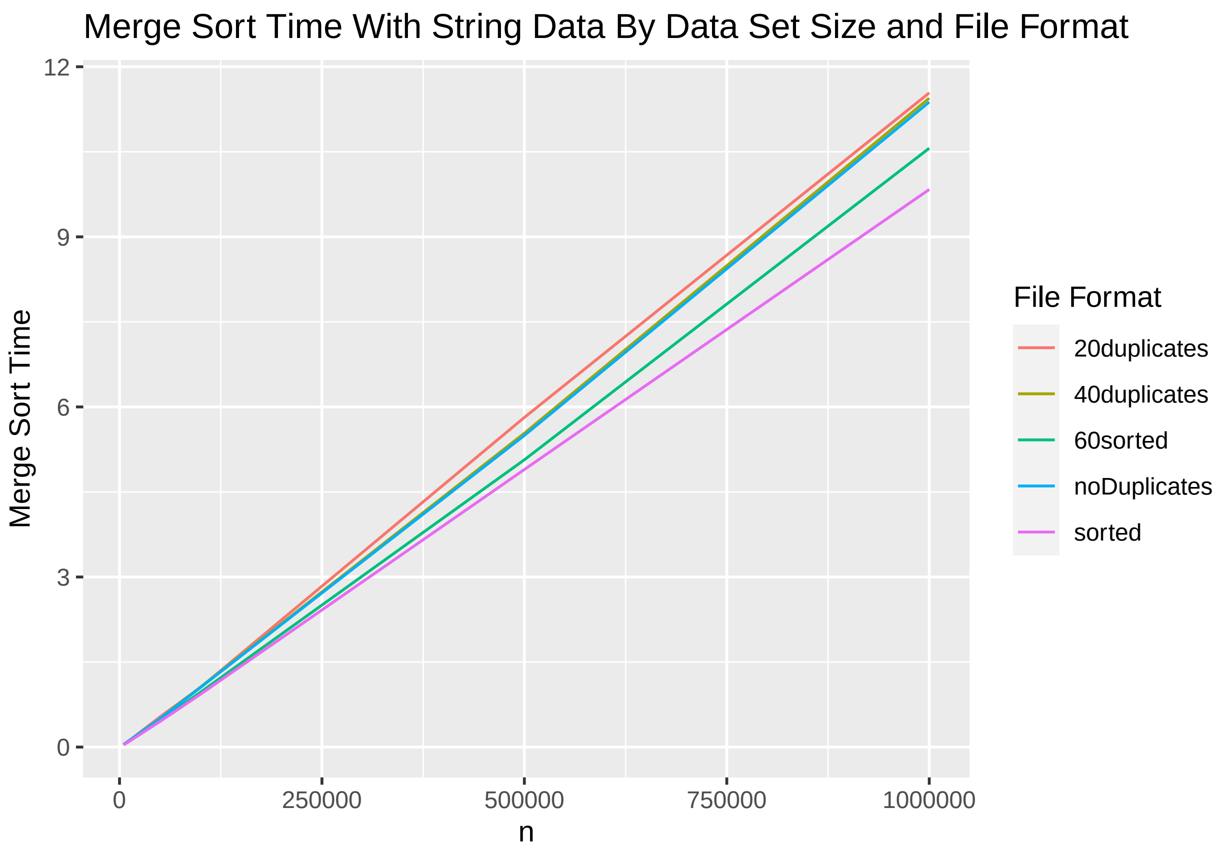
```
## `geom_smooth()` using formula 'y ~ x'
```



```
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"))
```

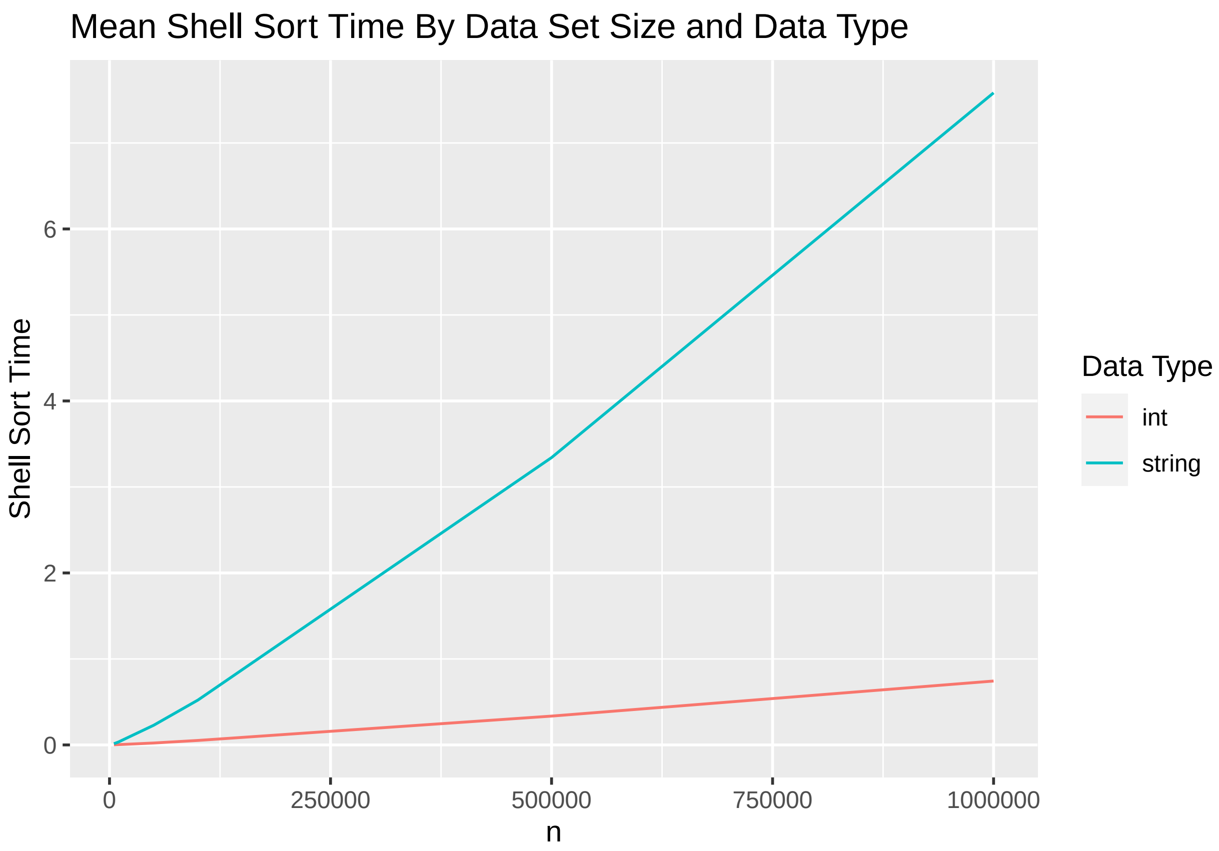


```
mergeStrings = subset(mergeTimes, var_type == "string")
ggplot(mergeStrings, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Merge
  guides(color = guide_legend(title = "File Format"))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



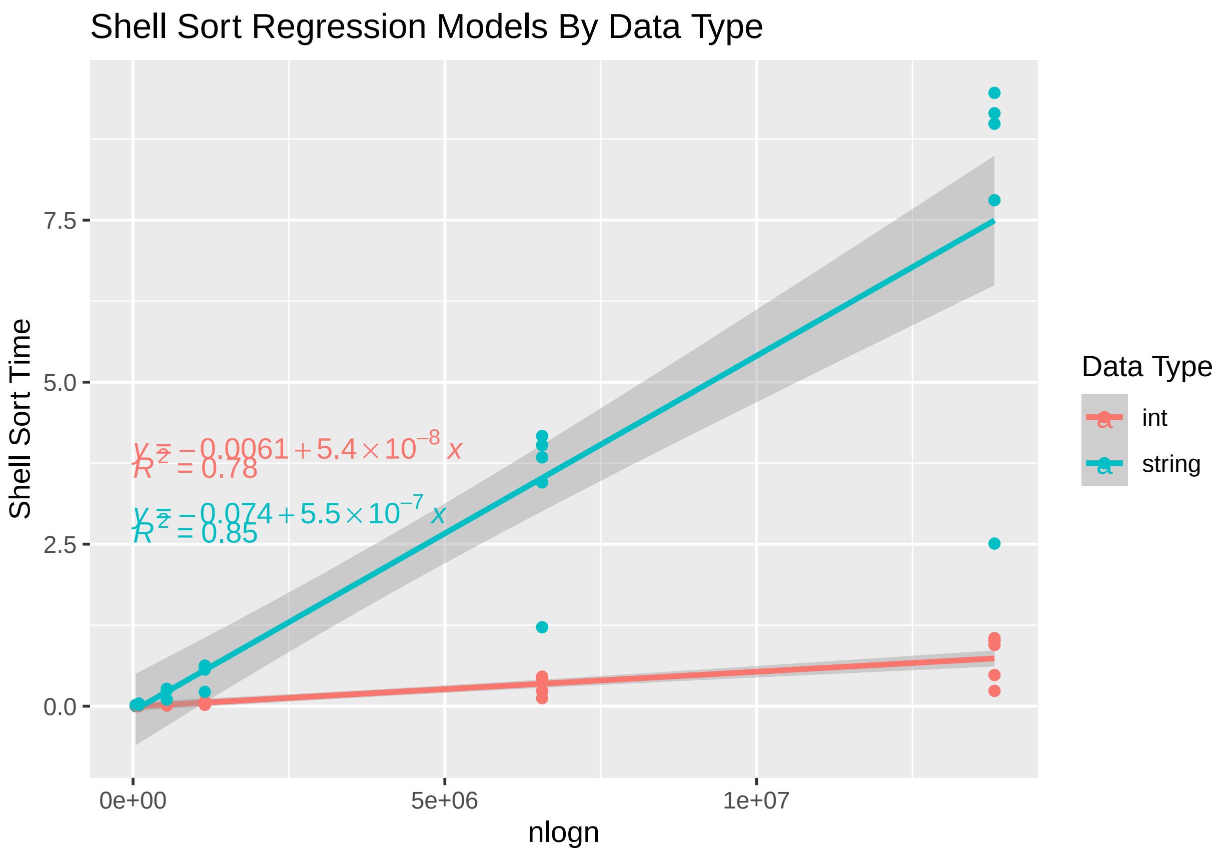
Shell Sort

```
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"))
```

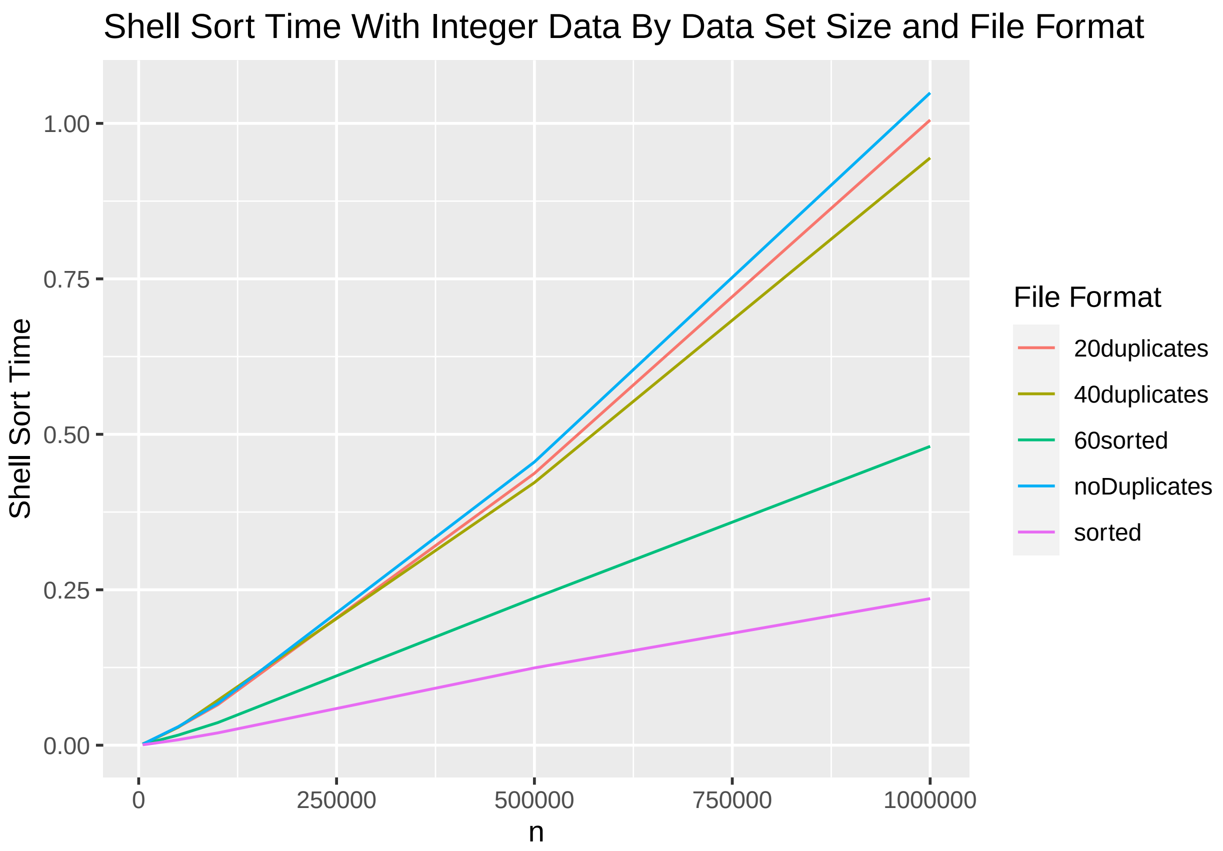


```
ggplot(shellTimes, aes(x = nlogn, y = shell_time, color = var_type)) +
  labs(title = "Shell Sort Regression Models By Data Type", x = "nlogn", y = "Shell Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

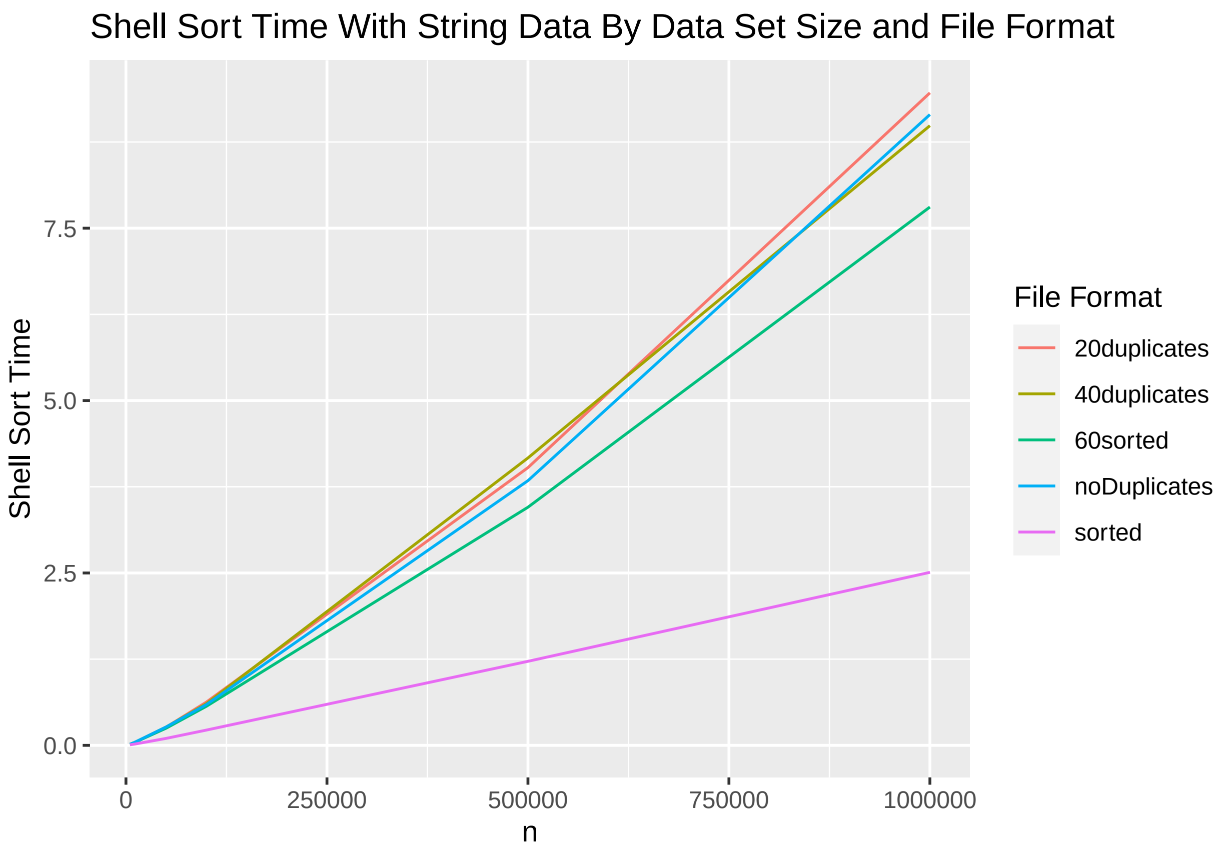
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



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

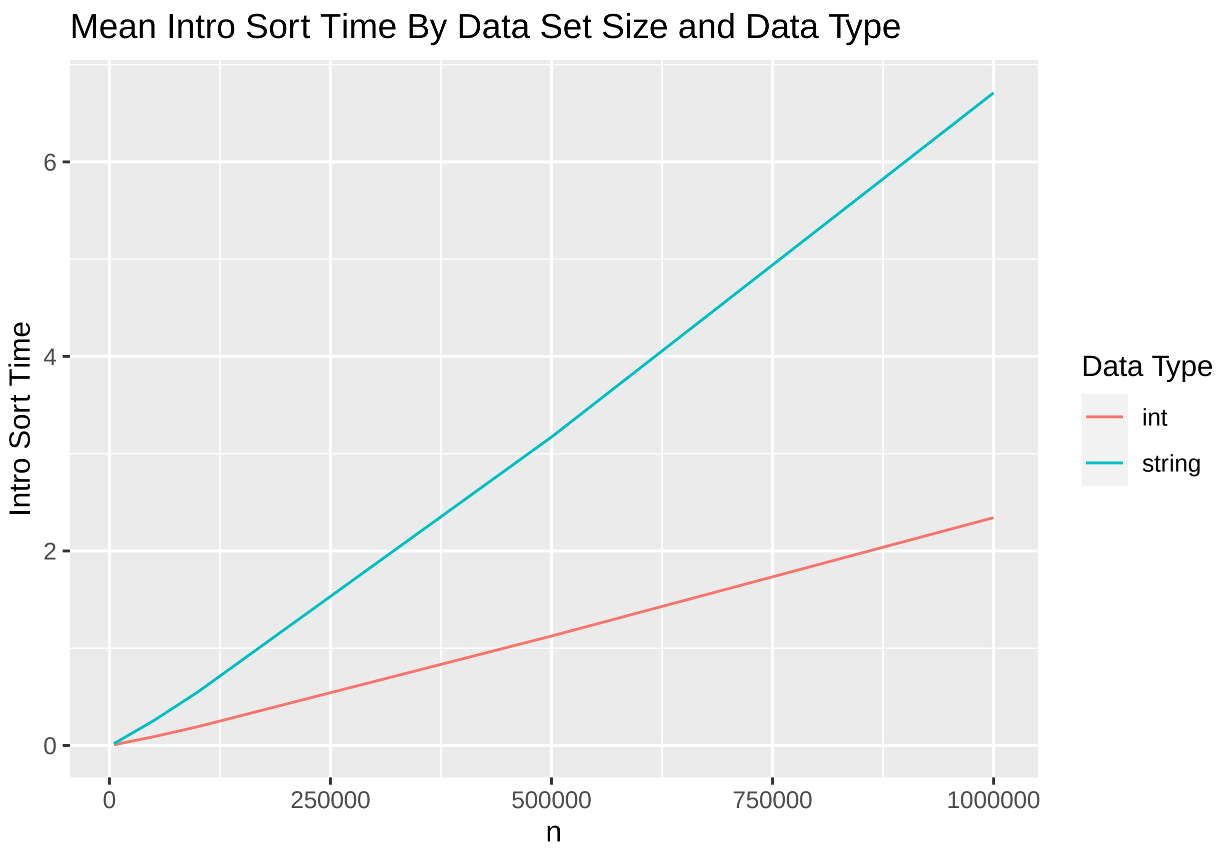


```
shellStrings = subset(shellTimes, var_type == "string")
ggplot(shellStrings, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Shell
  guides(color = guide_legend(title = "File Format"))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



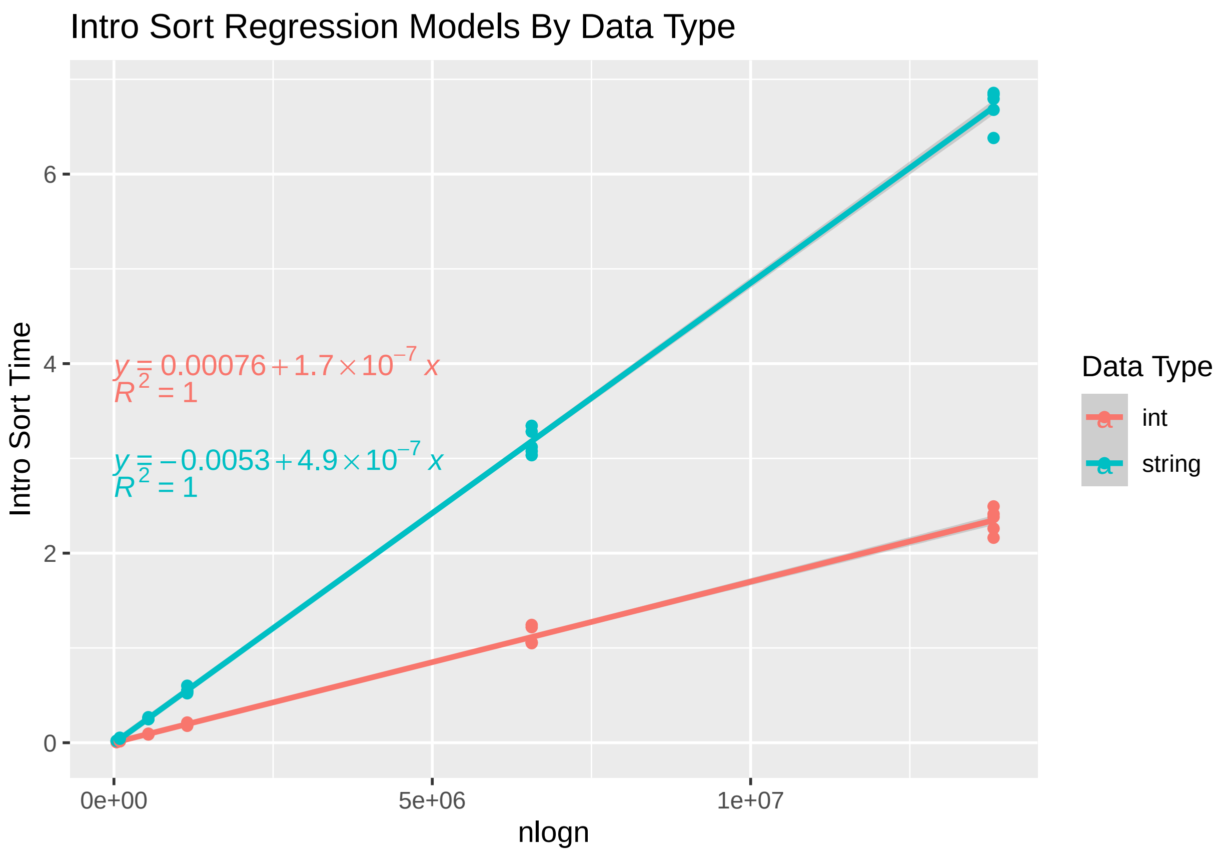
Intro Sort

```
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"))
```

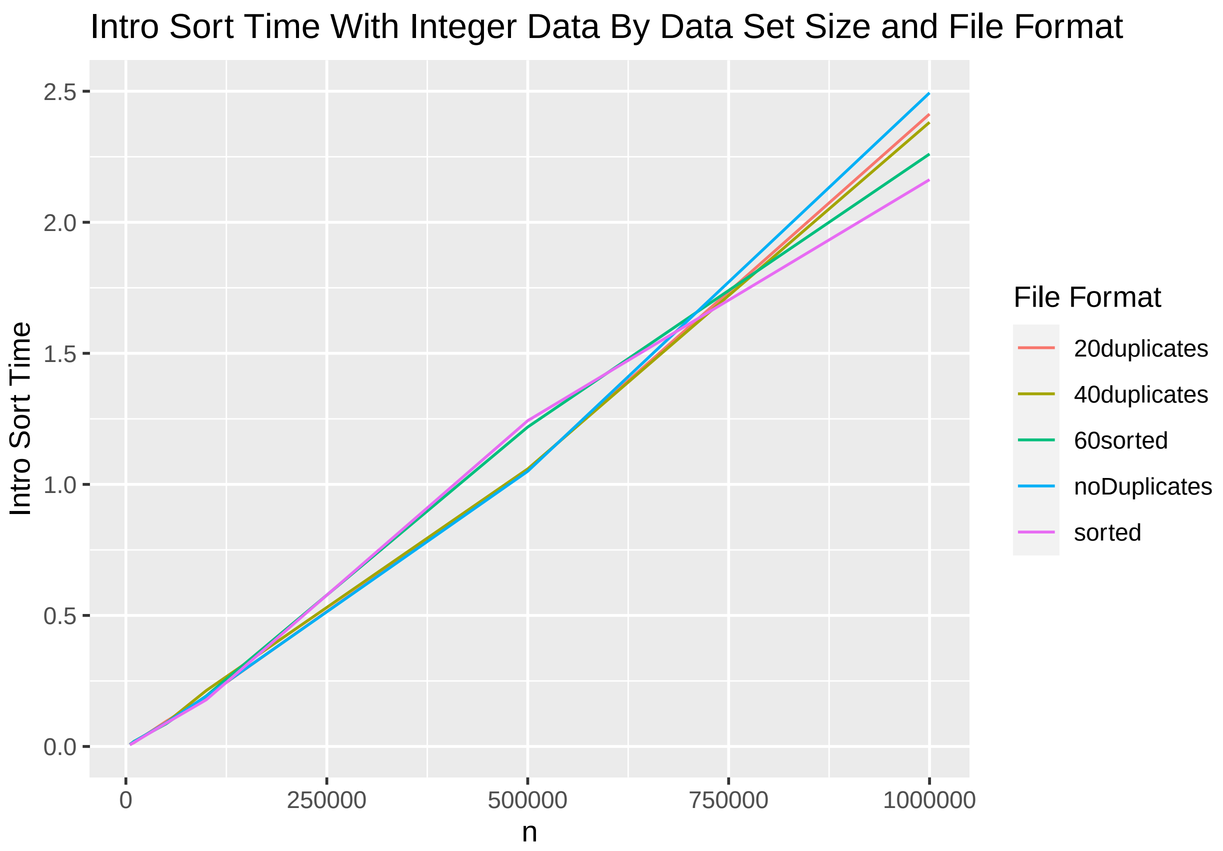


```
ggplot(introTimes, aes(x = nlogn, y = intro_time, color = var_type)) +
  labs(title = "Intro Sort Regression Models By Data Type", x = "nlogn", y = "Intro Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

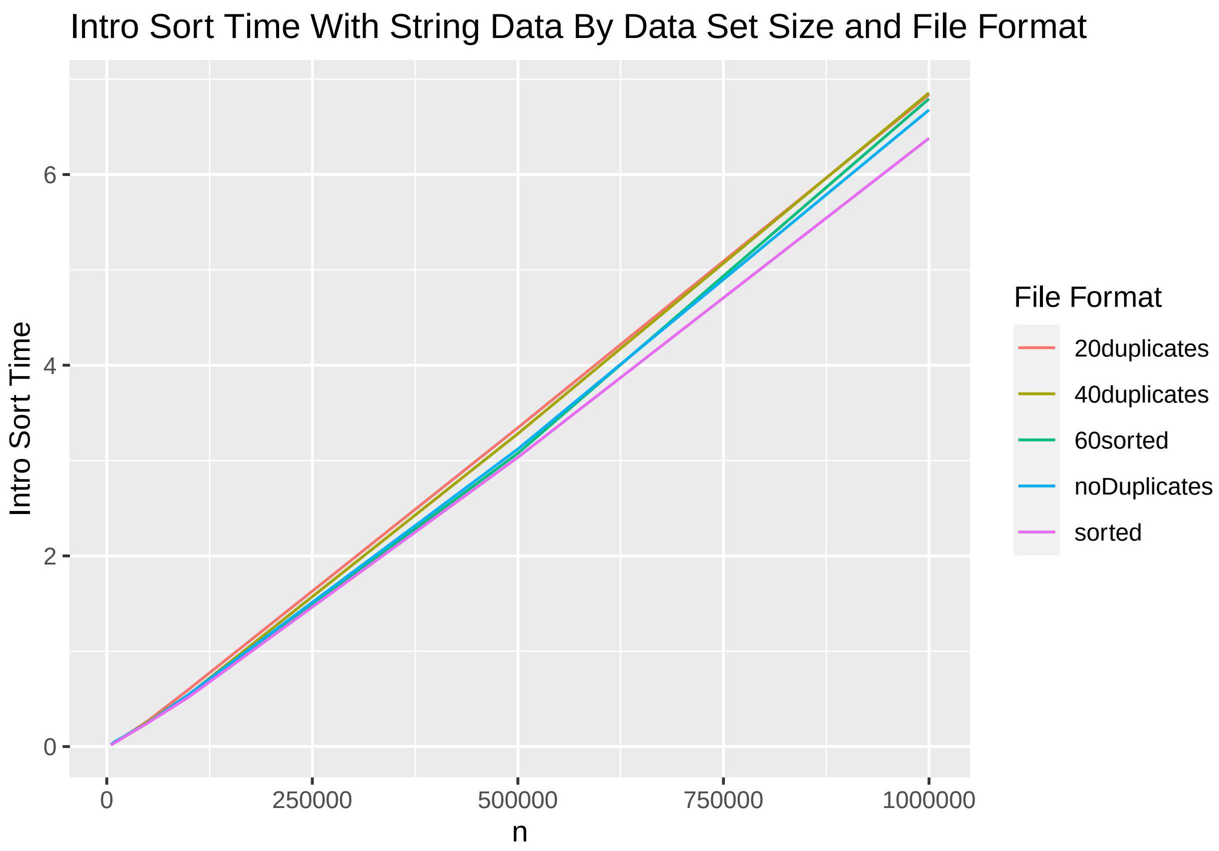
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

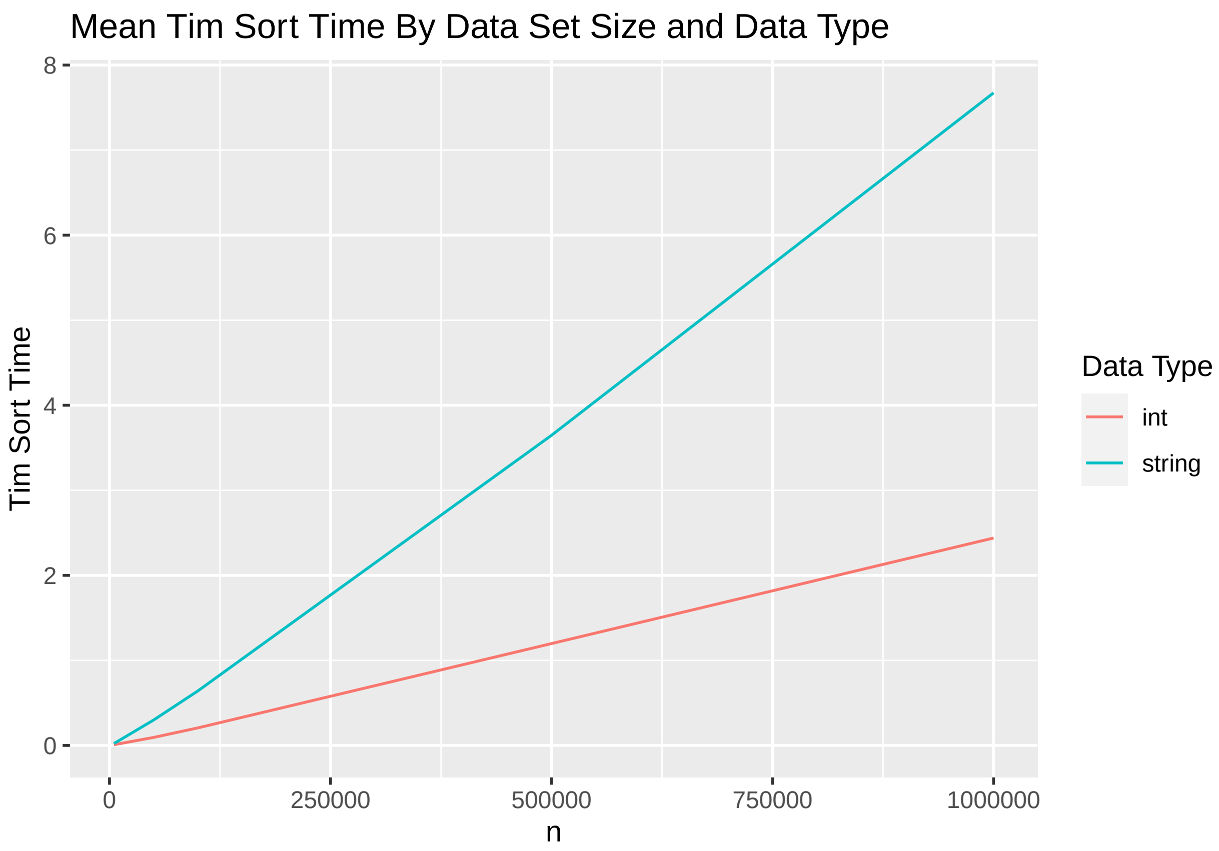


```
introStrings = subset(introTimes, var_type == "string")
ggplot(introStrings, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With String Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))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



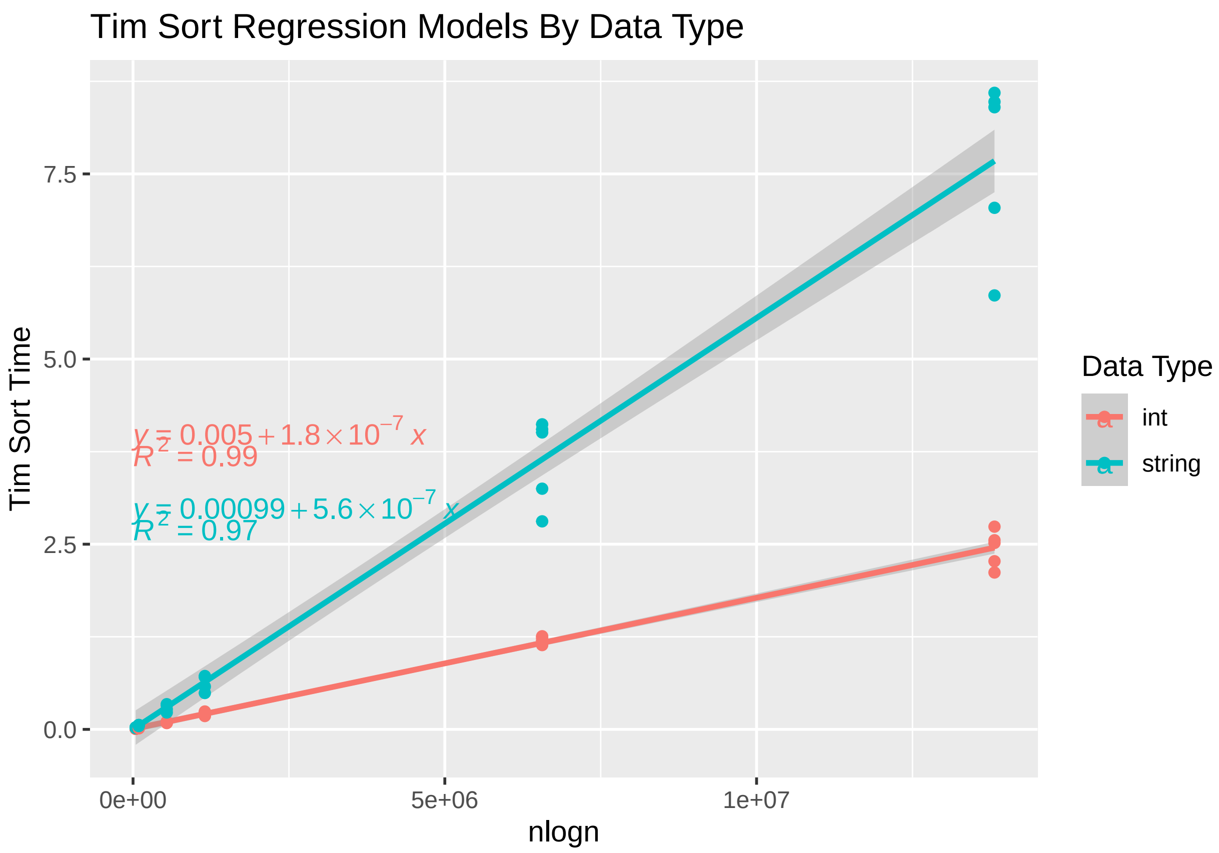
Tim Sort

```
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"))
```

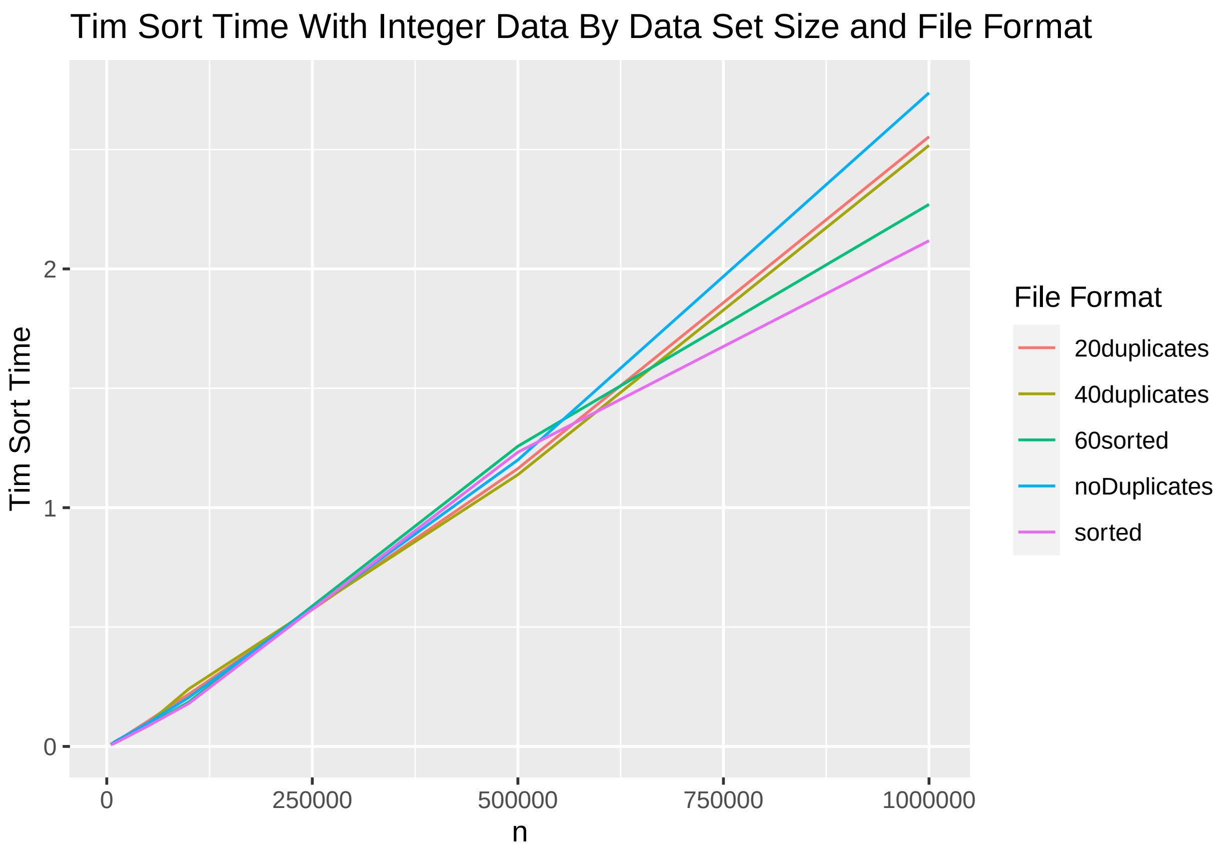


```
ggplot(timTimes, aes(x = nlogn, y = tim_time, color = var_type)) +
  labs(title = "Tim Sort Regression Models By Data Type", x = "nlogn", y = "Tim Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

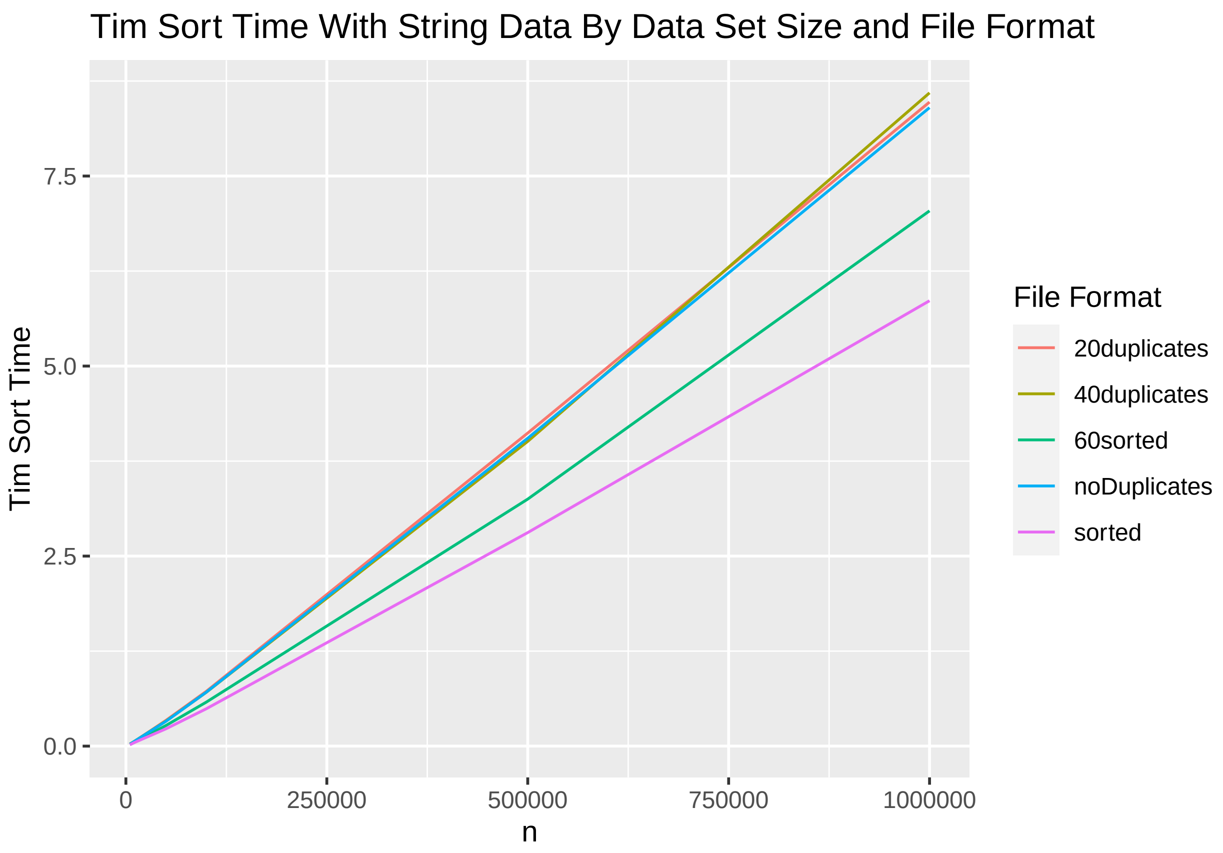
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```
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"))
```

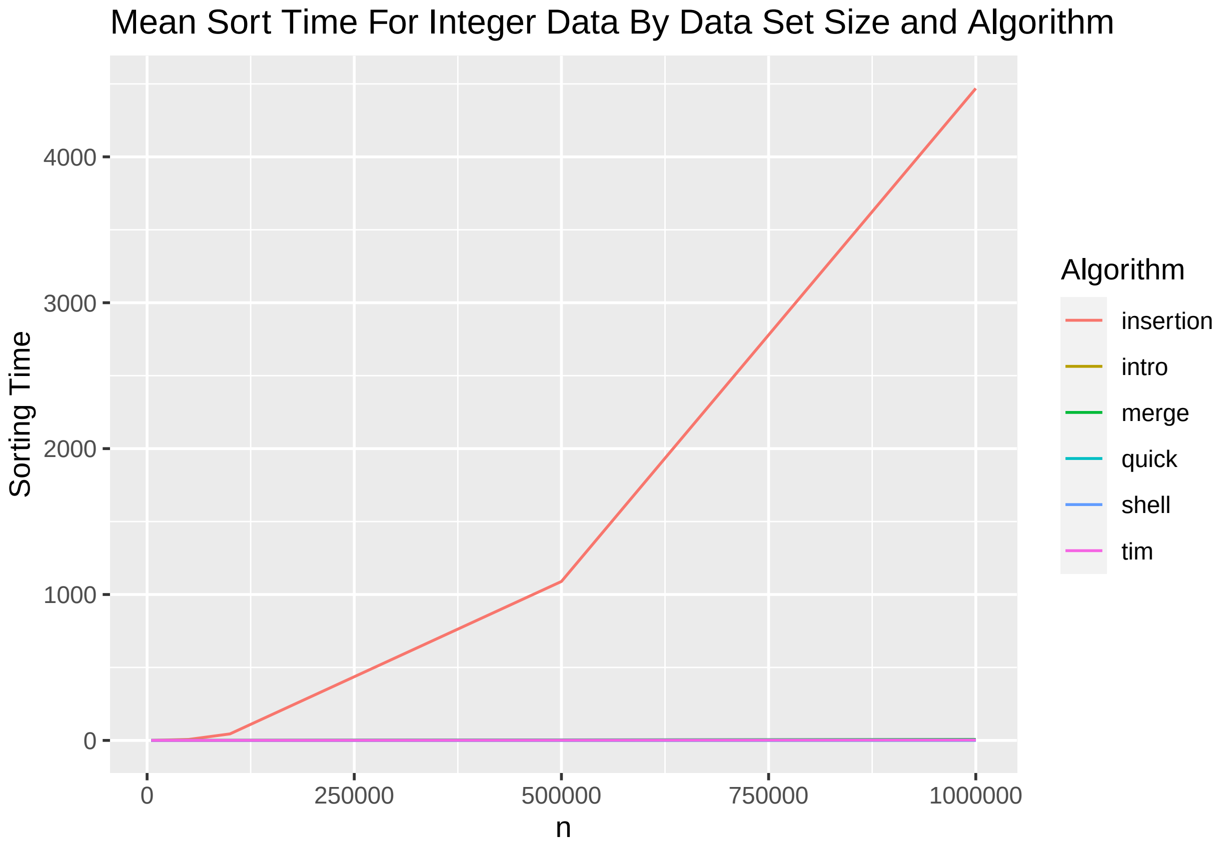


Algorithm Comparison

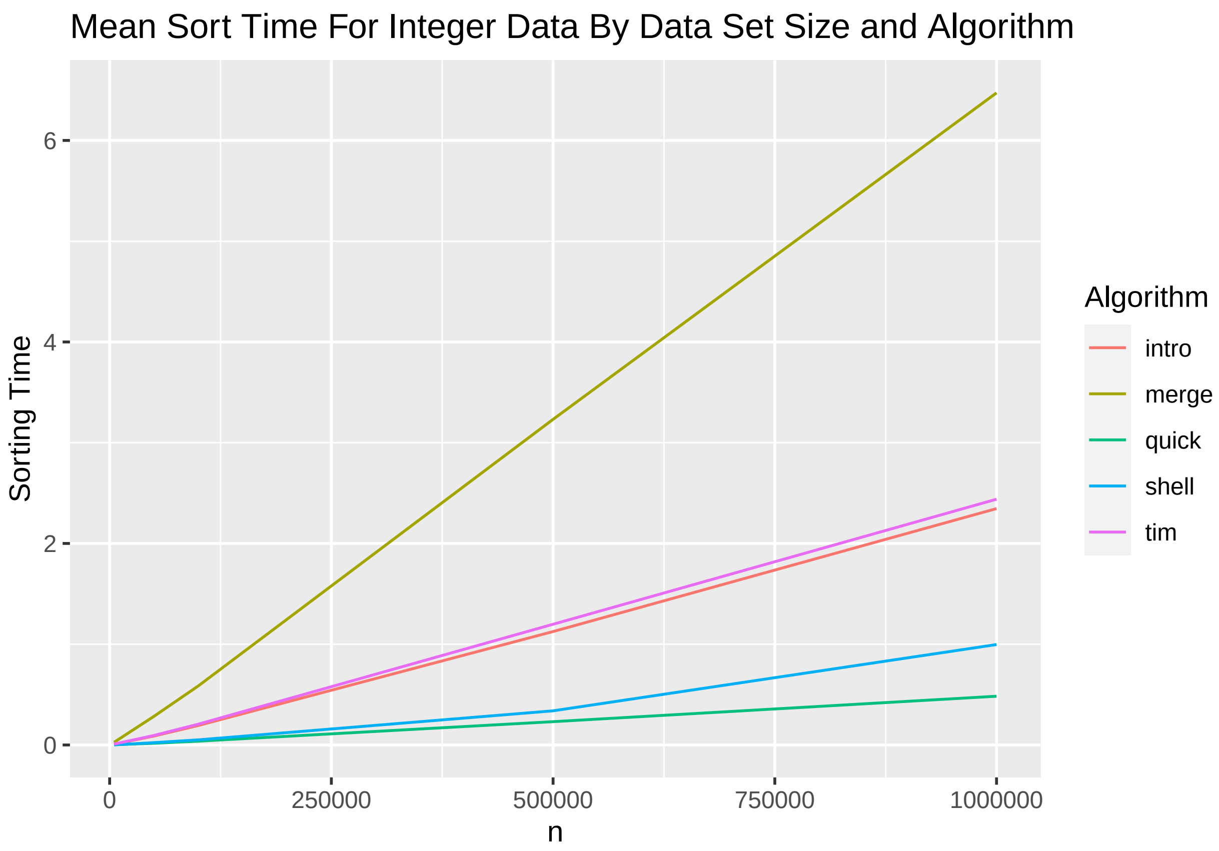
```
data2 = matrix(ncol = 5, nrow = 360)
for (i in 1:6) {
  for (j in 1:60) {
    data2[i * j, 1] = data[j, 1]
    data2[i * j, 2] = data[j, 2]
    data2[i * j, 3] = data[j, 3]
    data2[i * j, 4] = data[j, 3 + i]
    if (i == 1) {
      data2[i * j, 5] = "insertion"
    } else if (i == 2) {
      data2[i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

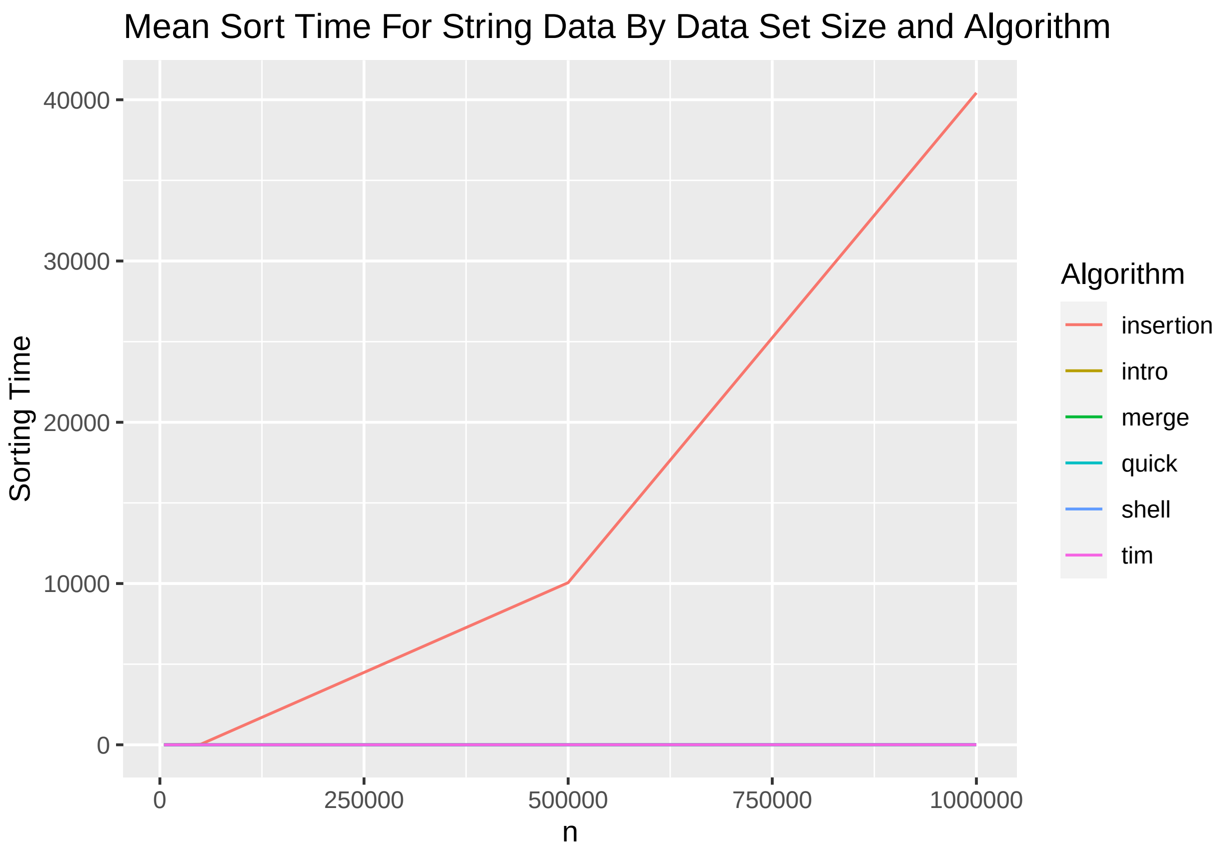
```
integerData = subset(data2, var_type == "int")
integerTimes = aggregate(time ~ algorithm + size, data = integerData, FUN = mean)
ggplot(integerTimes, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x =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



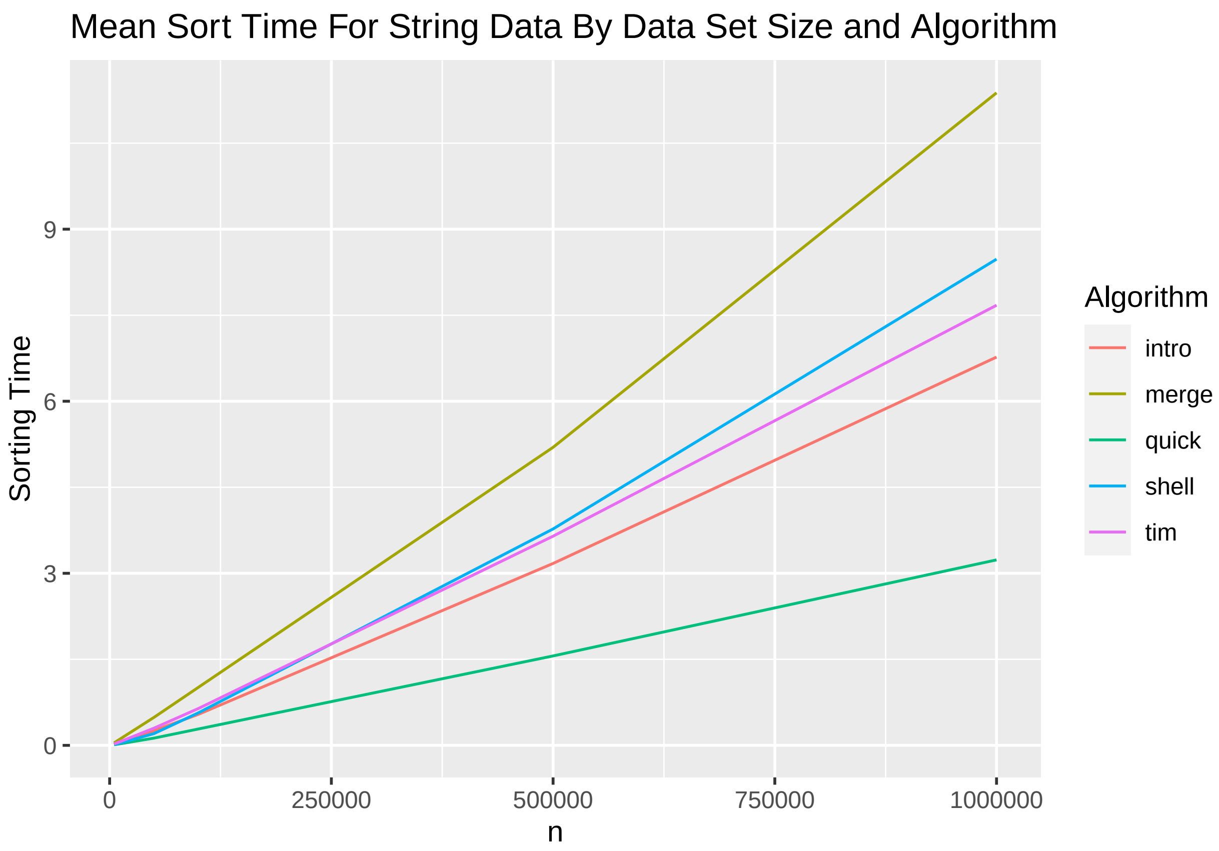
```
integerTimes2 = subset(integerTimes, algorithm != "insertion")
ggplot(integerTimes2, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x = "n", y = "Sorting 
  guides(color = guide_legend(title = "Algorithm"))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



```
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
0
3
6
9
0
250000
500000
750000
1000000
n
Algorithm
intro
merge
quick
shell
tim