Program 2 Graph Analysis

Ryan Schaefer and Wes Anderson

Create Dataset

```
library(ggplot2)
library(ggpubr)
data = read.csv("RyanDataRun1.csv")
data$n2 = data$size ^ 2
data$nlogn = log(data$size) * data$size
data
```

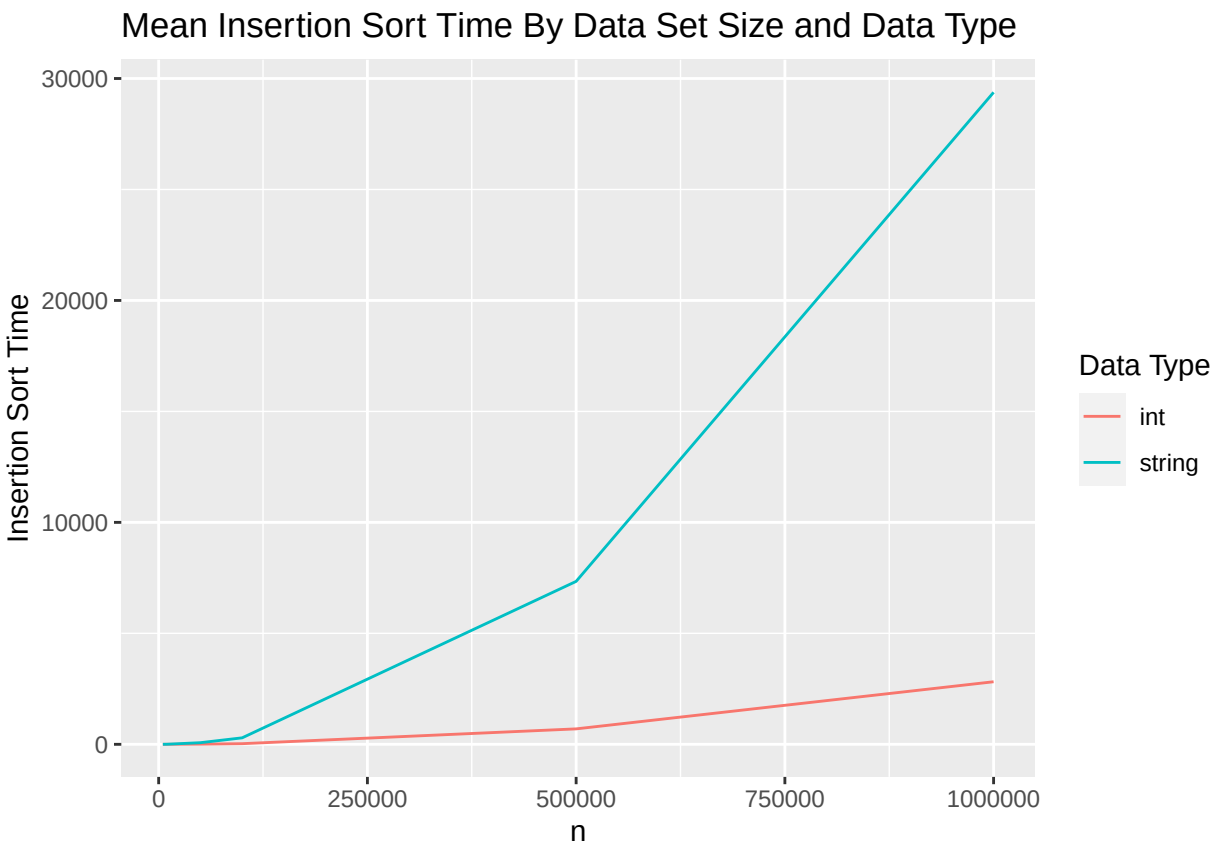
```
##    var_type    size       format insertion_time quick_time merge_time
## 1       int  500000 noDuplicates    1.06218e+03 0.20180800  2.7758600
## 2       int 1000000 40duplicates    4.42405e+03 0.46153300  6.2889400
## 3       int  100000 40duplicates    4.40678e+01 0.03881690  0.5888450
## 4       int   10000 40duplicates    4.52373e-01 0.00345996  0.0536523
## 5       int   50000       sorted    1.07108e-03 0.01395780  0.2825130
## 6       int   50000 20duplicates    1.11540e+01 0.01880440  0.2966160
## 7       int    5000 noDuplicates    1.13868e-01 0.00160935  0.0289623
## 8       int  500000       sorted    1.00746e-02 0.14355200  2.9121000
## 9       int  500000     60sorted    1.78701e+02 0.17067800  2.9877400
## 10      int   10000     60sorted    7.53939e-02 0.00256560  0.0532747
## 11      int 1000000 noDuplicates    4.46894e+03 0.44783300  6.1771600
## 12      int 1000000 20duplicates    4.48015e+03 0.44734800  6.2416800
## 13      int   50000 noDuplicates    1.12508e+01 0.01910750  0.2892180
## 14      int    5000     60sorted    1.80125e-02 0.00135961  0.0264883
## 15      int    5000       sorted    9.82800e-05 0.00127049  0.0265428
## 16      int  100000 20duplicates    4.44996e+01 0.03857260  0.5937180
## 17      int   50000     60sorted    1.79186e+00 0.01508750  0.2882440
## 18      int   10000 noDuplicates    4.51946e-01 0.00341646  0.0539085
## 19      int  500000 20duplicates    1.11640e+03 0.21144900  3.0382400
## 20      int  500000 40duplicates    1.11854e+03 0.21542200  3.0373700
## 21      int 1000000       sorted    2.21053e-02 0.35156800  5.9576300
## 22      int    5000 20duplicates    1.11112e-01 0.00163077  0.0272923
## 23      int  100000 noDuplicates    4.45871e+01 0.03824000  0.6001480
## 24      int   50000 40duplicates    1.11186e+01 0.02000380  0.2842830
## 25      int   10000 20duplicates    4.48142e-01 0.00322118  0.0556267
## 26      int  100000       sorted    1.96880e-03 0.02661890  0.5728220
## 27      int  100000     60sorted    7.15581e+00 0.02993020  0.5773300
## 28      int   10000       sorted    2.00159e-04 0.00273008  0.0529762
## 29      int    5000 40duplicates    1.11897e-01 0.00170964  0.0260556
## 30      int 1000000     60sorted    7.15488e+02 0.39774900  6.0698300
## 31   string   50000       sorted    8.79662e-03 0.13973700  0.4729300
## 32   string  500000 20duplicates    1.02101e+04 1.64007000  5.8209500
## 33   string   50000 20duplicates    1.01569e+02 0.12311400  0.5483400
```

```
## 34   string   10000 40duplicates    4.08111e+00 0.02101890  0.0993715
## 35   string   10000     60sorted    2.59330e+00 0.02274090  0.0931251
## 36   string  100000       sorted    1.85365e-02 0.24824700  0.9689940
## 37   string    5000 40duplicates    9.99780e-01 0.00961879  0.0467183
## 38   string  500000     60sorted    6.55264e+03 1.52423000  4.9884700
## 39   string   50000 noDuplicates    9.61621e+01 0.12617400  0.4838120
## 40   string  500000 40duplicates    9.88785e+03 1.54019000  5.6256000
## 41   string    5000 20duplicates    9.81218e-01 0.01021840  0.0444044
## 42   string  100000 noDuplicates    3.95303e+02 0.27156800  1.0583400
## 43   string    5000 noDuplicates    9.94519e-01 0.00873819  0.0448532
## 44   string  100000     60sorted    2.53332e+02 0.26832700  0.9953050
## 45   string 1000000 20duplicates    4.01252e+04 3.29134000 11.8904000
## 46   string   10000 noDuplicates    4.01830e+00 0.02266350  0.1001880
## 47   string 1000000 noDuplicates    4.04289e+04 3.27891000 11.7127000
## 48   string 1000000       sorted    1.84133e-01 3.13208000 10.1129000
## 49   string  500000 noDuplicates    1.00548e+04 1.50580000  5.6929300
## 50   string  100000 40duplicates    4.00866e+02 0.28591100  1.0802000
## 51   string    5000     60sorted    6.27955e-01 0.00991239  0.0447340
## 52   string 1000000     60sorted    2.57302e+04 3.29847000 11.0195000
## 53   string    5000       sorted    8.75481e-04 0.00940053  0.0411268
## 54   string  100000 20duplicates    4.08117e+02 0.27068000  1.0982000
## 55   string   10000 20duplicates    4.05438e+00 0.02055750  0.0961895
## 56   string   10000       sorted    1.84725e-03 0.02161620  0.0844152
## 57   string  500000       sorted    9.52993e-02 1.62471000  5.0404400
## 58   string   50000 40duplicates    1.01896e+02 0.13606500  0.5267900
## 59   string   50000     60sorted    6.58944e+01 0.13759900  0.4847410
## 60   string 1000000 40duplicates    4.05995e+04 3.24490000 11.7488000
##    shell_time intro_time   tim_time      n2       nlogn
## 1  0.40866000 0.99389500 1.09327000 2.5e+11  6561181.69
## 2  0.95685700 2.30195000 2.51588000 1.0e+12 13815510.56
## 3  0.06687330 0.19398700 0.21268500 1.0e+10  1151292.55
## 4  0.00454805 0.01614760 0.01727000 1.0e+08    92103.40
## 5  0.00889380 0.08678940 0.08480020 2.5e+09   540988.91
## 6  0.02922770 0.09039230 0.11047700 2.5e+09   540988.91
## 7  0.00187175 0.00776139 0.00857962 2.5e+07    42585.97
## 8  0.11139400 1.07620000 1.01744000 2.5e+11  6561181.69
## 9  0.22225400 1.07879000 1.10298000 2.5e+11  6561181.69
## 10 0.00275612 0.01637040 0.01528680 1.0e+08    92103.40
## 11 0.97348700 2.39373000 2.51168000 1.0e+12 13815510.56
## 12 0.99476000 2.33022000 2.48783000 1.0e+12 13815510.56
## 13 0.03058590 0.09222580 0.10039500 2.5e+09   540988.91
## 14 0.00120756 0.00642091 0.00688942 2.5e+07    42585.97
## 15 0.00067370 0.00628704 0.00739910 2.5e+07    42585.97
## 16 0.06671380 0.19373500 0.21538600 1.0e+10  1151292.55
## 17 0.01625780 0.08952010 0.09035670 2.5e+09   540988.91
## 18 0.00447603 0.01640210 0.01751760 1.0e+08    92103.40
## 19 0.43633300 1.11411000 1.17721000 2.5e+11  6561181.69
## 20 0.43611900 1.11028000 1.19260000 2.5e+11  6561181.69
## 21 0.23740700 2.27368000 2.14699000 1.0e+12 13815510.56
## 22 0.00182527 0.00717432 0.00791560 2.5e+07    42585.97
## 23 0.06749090 0.19803400 0.21700200 1.0e+10  1151292.55
## 24 0.02924930 0.09264560 0.09926380 2.5e+09   540988.91
## 25 0.00418070 0.01461220 0.01960080 1.0e+08    92103.40
## 26 0.02029610 0.18338300 0.18512500 1.0e+10  1151292.55
```

```
## 27 0.03831370 0.19717300 0.19623500 1.0e+10  1151292.55
## 28 0.00147358 0.01798480 0.01384390 1.0e+08    92103.40
## 29 0.00207455 0.00907821 0.00798345 2.5e+07    42585.97
## 30 0.47795900 2.21196000 2.28335000 1.0e+12 13815510.56
## 31 0.10246600 0.26962200 0.23438500 2.5e+09   540988.91
## 32 3.96140000 3.40174000 4.18839000 2.5e+11  6561181.69
## 33 0.27166200 0.27445500 0.35617200 2.5e+09   540988.91
## 34 0.03872480 0.04357720 0.06105450 1.0e+08    92103.40
## 35 0.03660720 0.04861210 0.04987010 1.0e+08    92103.40
## 36 0.21728100 0.56300200 0.51685400 1.0e+10  1151292.55
## 37 0.01667430 0.01917410 0.02803100 2.5e+07    42585.97
## 38 3.44751000 2.97456000 3.15880000 2.5e+11  6561181.69
## 39 0.25822100 0.25354200 0.31277400 2.5e+09   540988.91
## 40 4.24201000 3.29229000 4.09191000 2.5e+11  6561181.69
## 41 0.01662610 0.02328790 0.02562460 2.5e+07    42585.97
## 42 0.59968300 0.58061000 0.72557600 1.0e+10  1151292.55
## 43 0.01662950 0.01939320 0.02561510 2.5e+07    42585.97
## 44 0.57446600 0.53799300 0.59324300 1.0e+10  1151292.55
## 45 9.78971000 7.16212000 8.86707000 1.0e+12 13815510.56
## 46 0.04093270 0.06971220 0.06806260 1.0e+08    92103.40
## 47 9.21126000 6.71851000 8.74064000 1.0e+12 13815510.56
## 48 2.55454000 6.68877000 6.08425000 1.0e+12 13815510.56
## 49 3.97561000 3.26302000 4.16314000 2.5e+11  6561181.69
## 50 0.62235800 0.57354300 0.73692000 1.0e+10  1151292.55
## 51 0.01453950 0.01959530 0.02128110 2.5e+07    42585.97
## 52 8.11616000 7.01413000 7.29371000 1.0e+12 13815510.56
## 53 0.00870018 0.01819550 0.01895180 2.5e+07    42585.97
## 54 0.64909900 0.56754000 0.75672300 1.0e+10  1151292.55
## 55 0.03783020 0.04363100 0.05995260 1.0e+08    92103.40
## 56 0.01730530 0.04510620 0.04377140 1.0e+08    92103.40
## 57 1.25441000 3.28741000 2.98305000 2.5e+11  6561181.69
## 58 0.27059600 0.27321300 0.34293000 2.5e+09   540988.91
## 59 0.26119400 0.25779400 0.28684900 2.5e+09   540988.91
## 60 9.27403000 7.18690000 8.88903000 1.0e+12 13815510.56
```

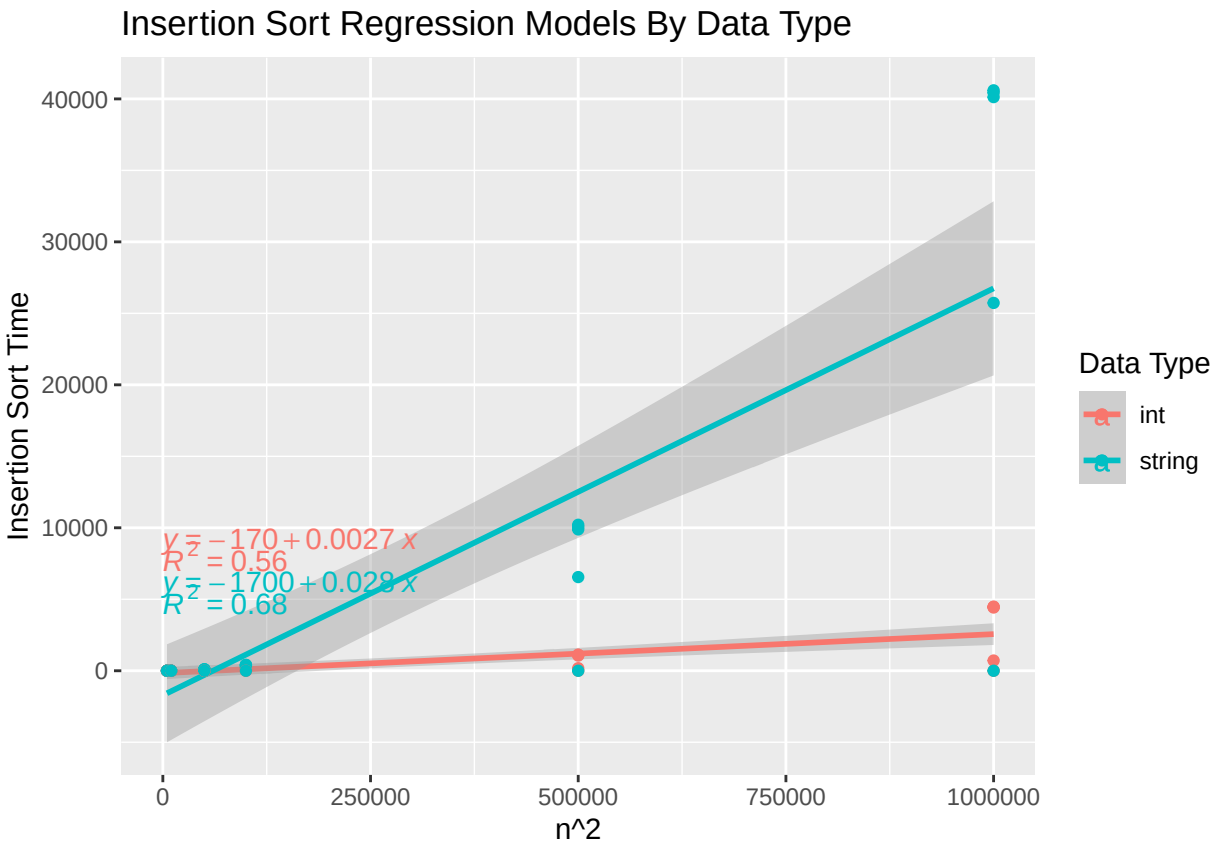
Insertion Sort

```
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"))
```

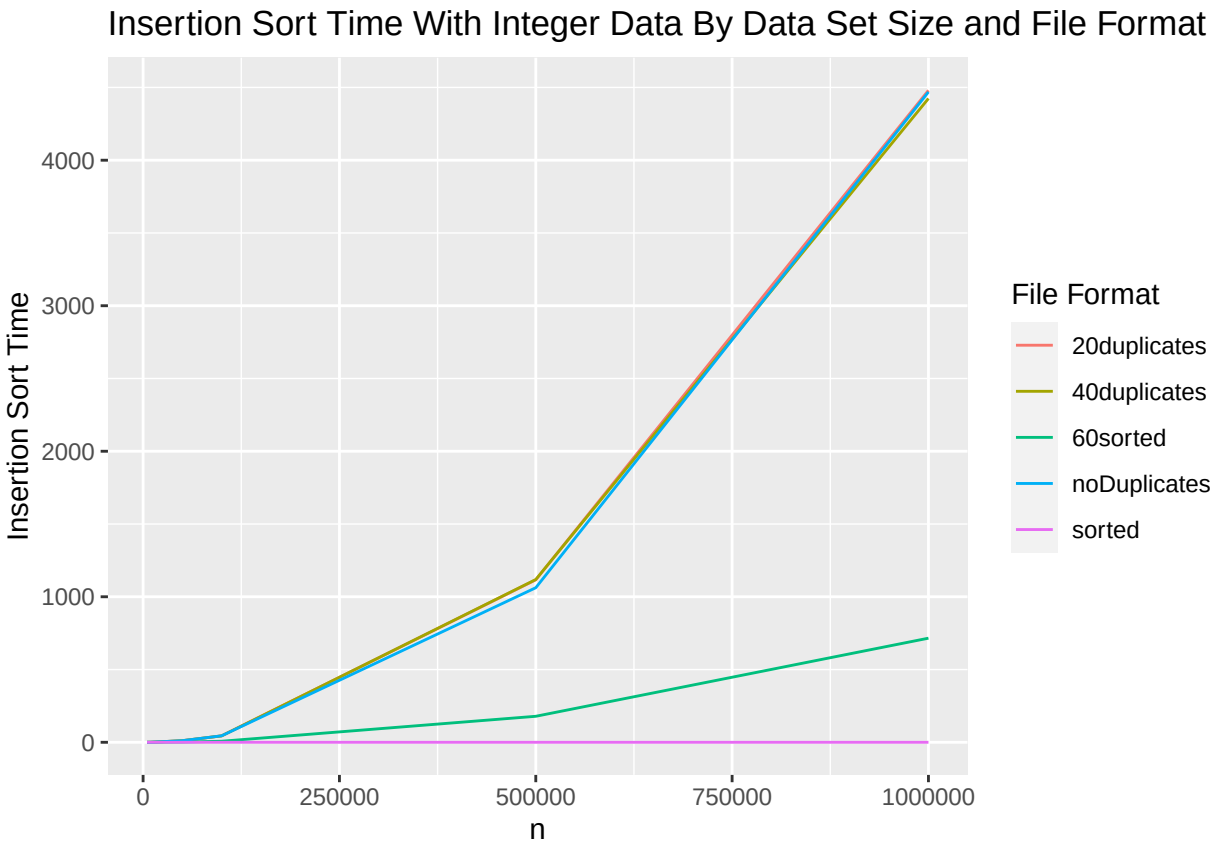



```
ggplot(insertionTimes, aes(x = size, y = insertion_time, color = var_type)) +
  labs(title = "Insertion Sort Regression Models By Data Type", x = "n^2", y = "Insertion Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(9000, 6000)) +
  stat_cor(aes(label=..rr.label..),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

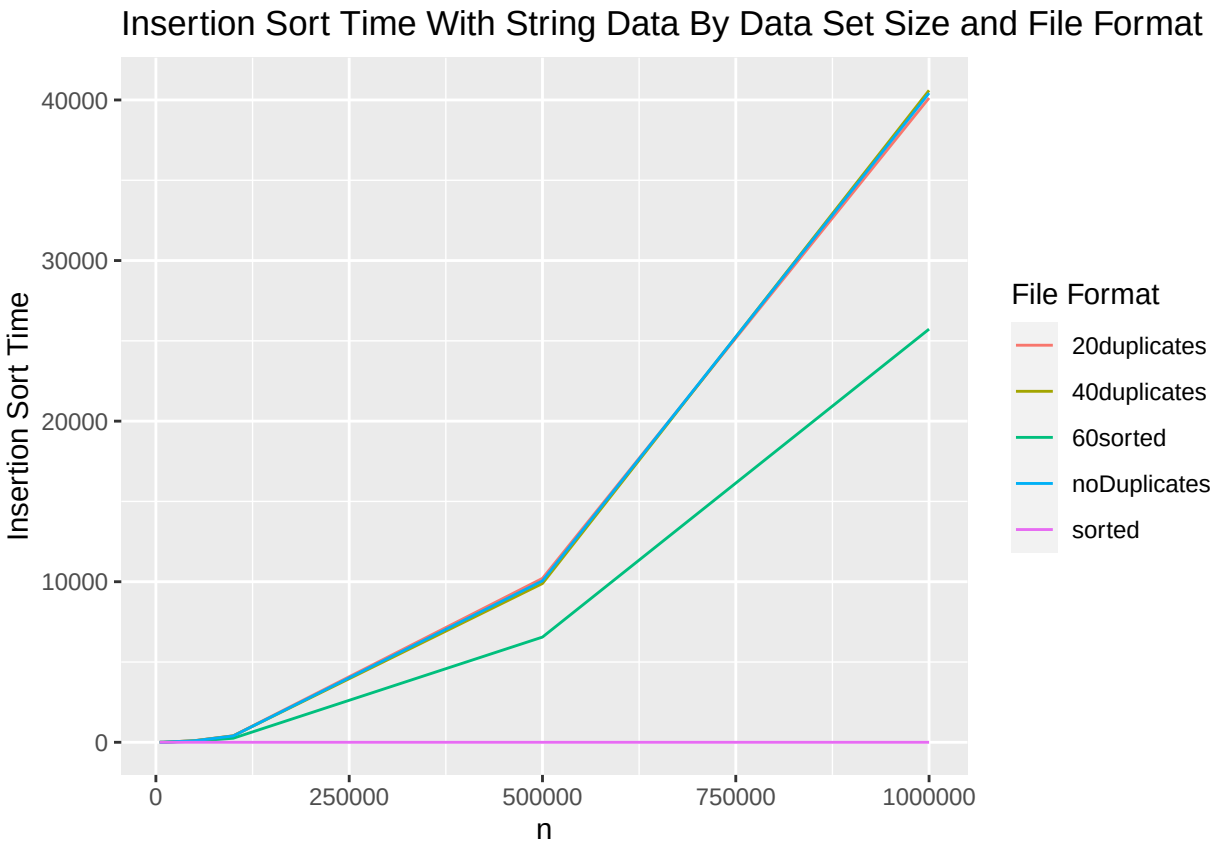
```
## `geom_smooth()` using formula 'y ~ x'
```




```
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"))
```

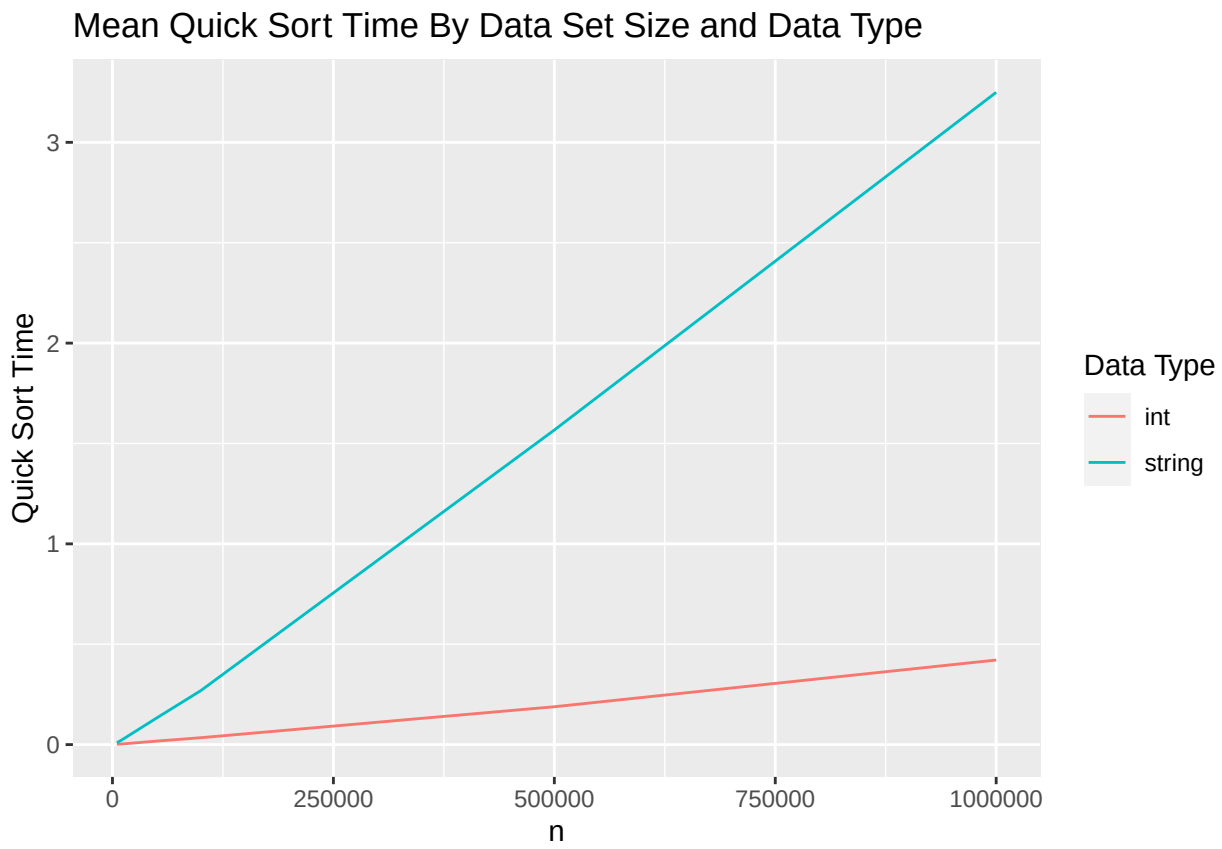



```
insertionStrings = subset(insertionTimes, var_type == "string")
ggplot(insertionStrings, aes(x = size, y = insertion_time, color = format)) +
  geom_line() +
  labs(title = "Insertion Sort Time With String Data By Data Set Size and File Format", x = "n", y = "I
  guides(color = guide_legend(title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



Quick Sort

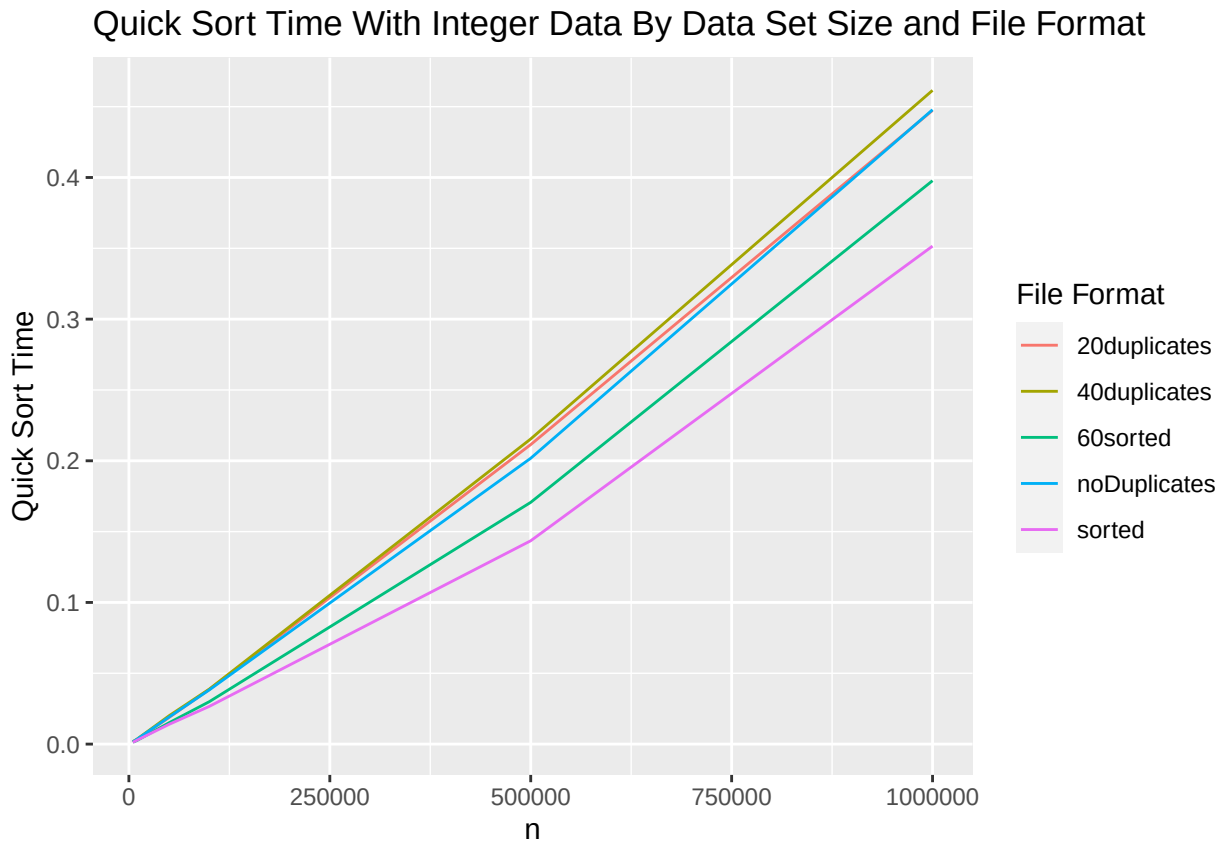
```
quickTimes = aggregate(quick_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
quickTimes2 = aggregate(quick_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(quickTimes2, aes(x = size, y = quick_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Quick Sort 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'
```

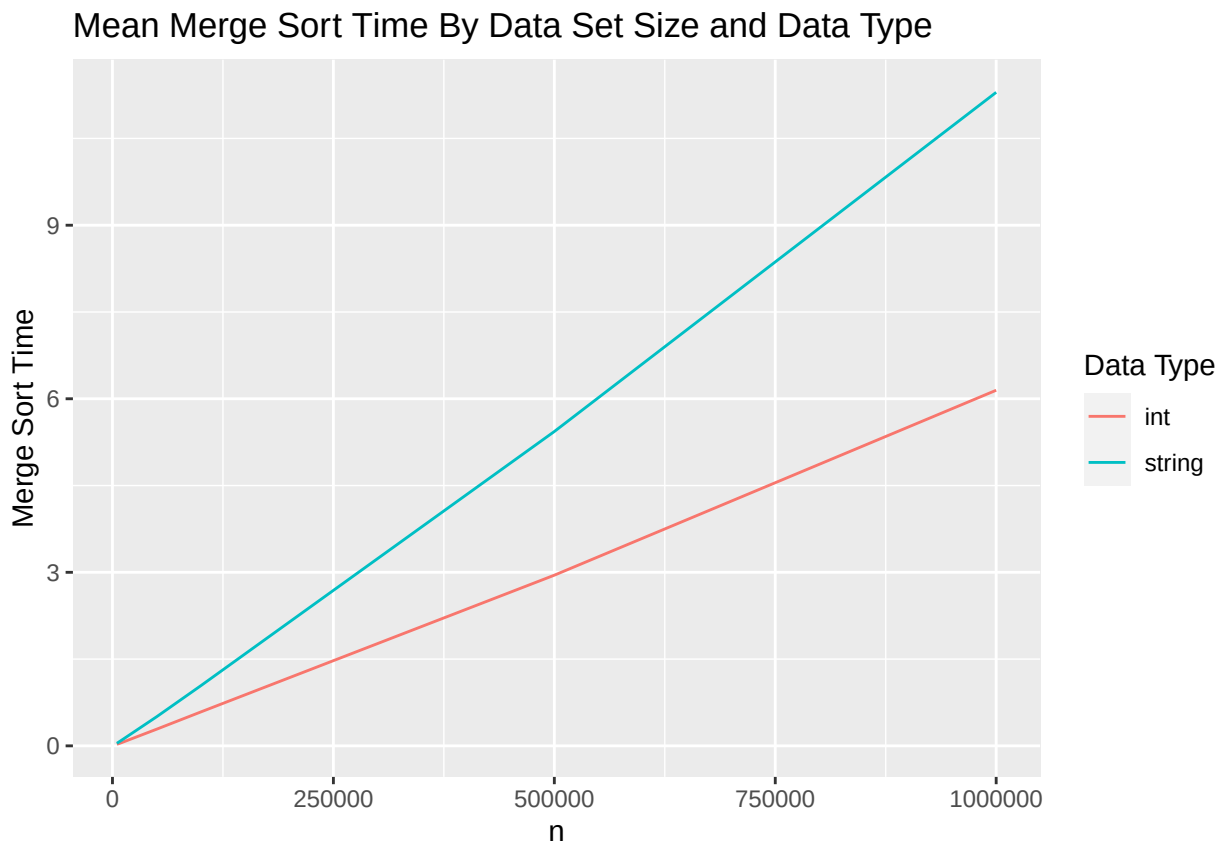

```
quickInts = subset(quickTimes, var_type == "int")
ggplot(quickInts, aes(x = size, y = quick_time, color = format)) +
  geom_line() +
  labs(title = "Quick Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Quick
  guides(color = guide_legend(title = "File Format"))
```




```
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"))
```

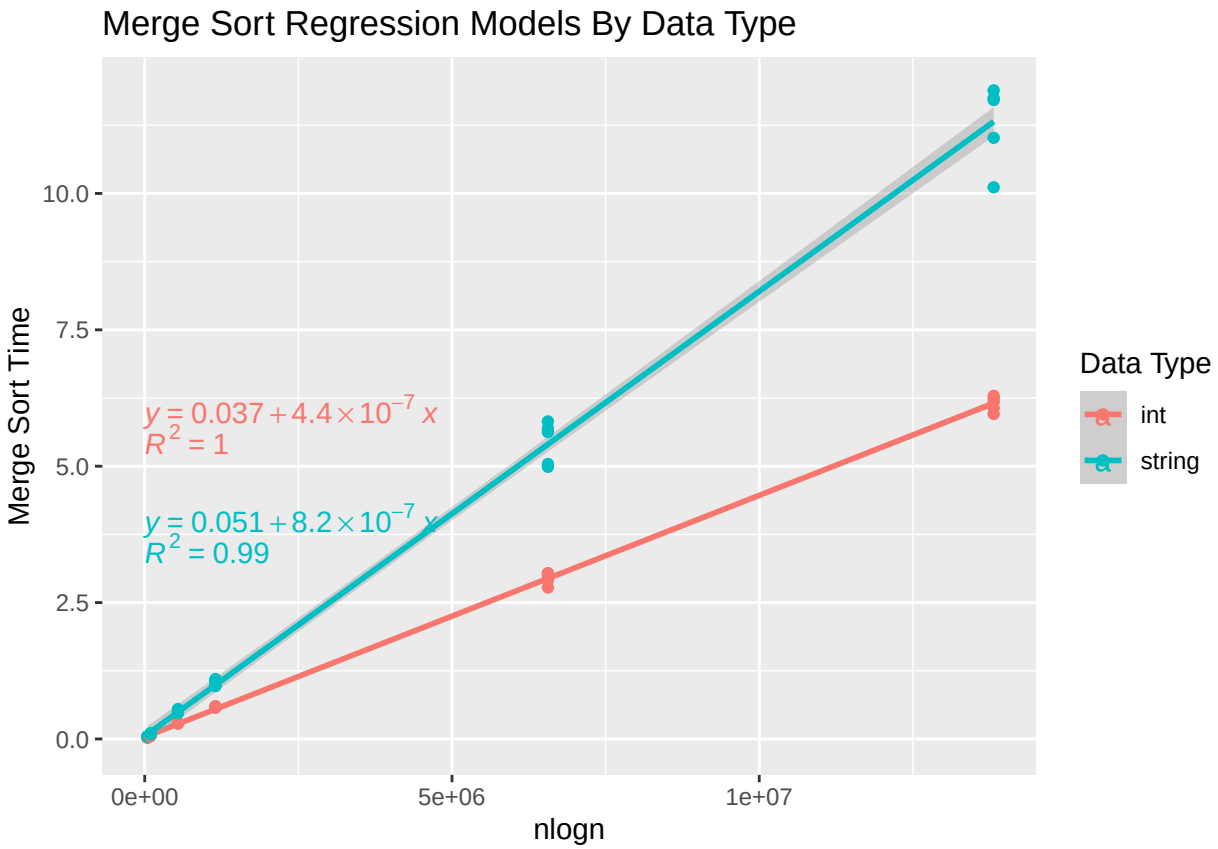
Merge Sort

```
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"))
```

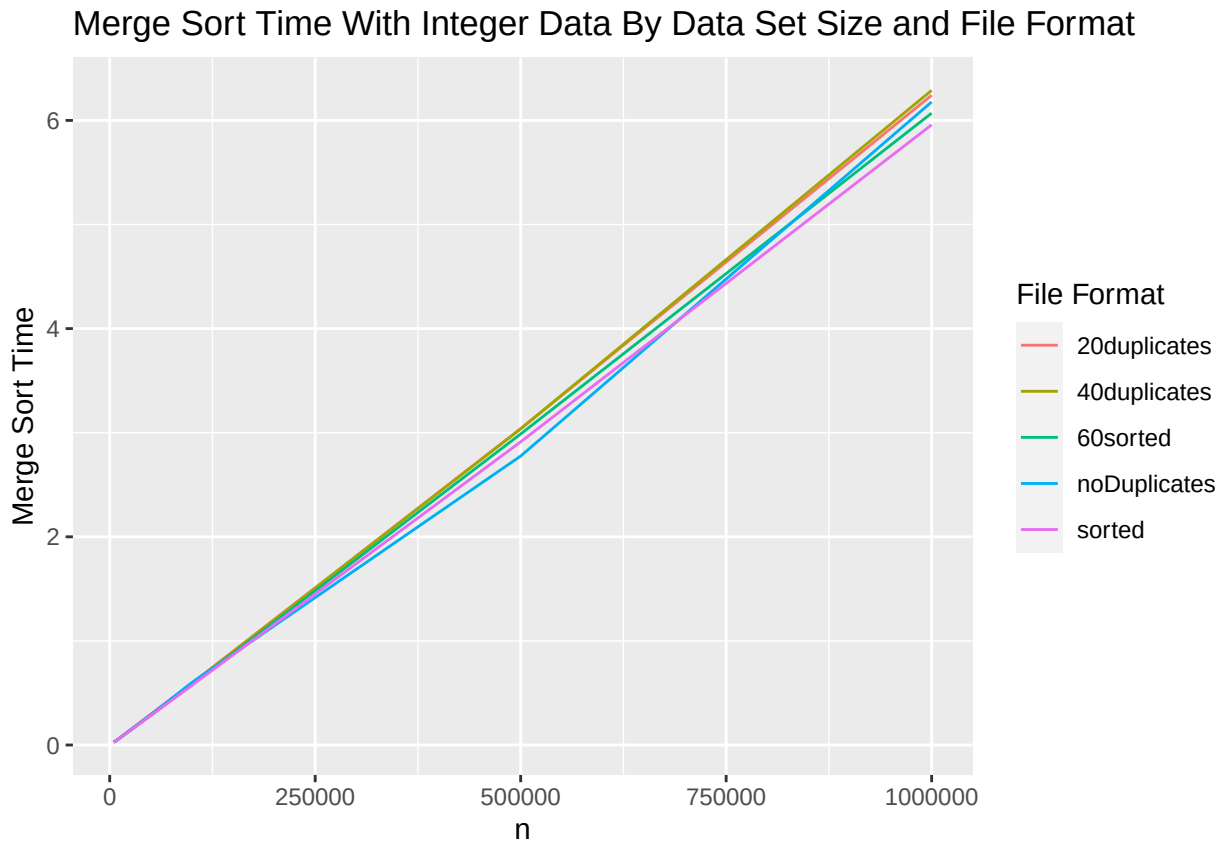



```
ggplot(mergeTimes, aes(x = nlogn, y = merge_time, color = var_type)) +
  labs(title = "Merge Sort Regression Models By Data Type", x = "nlogn", y = "Merge Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(6, 4)) +
  stat_cor(aes(label=..rr.label..),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

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



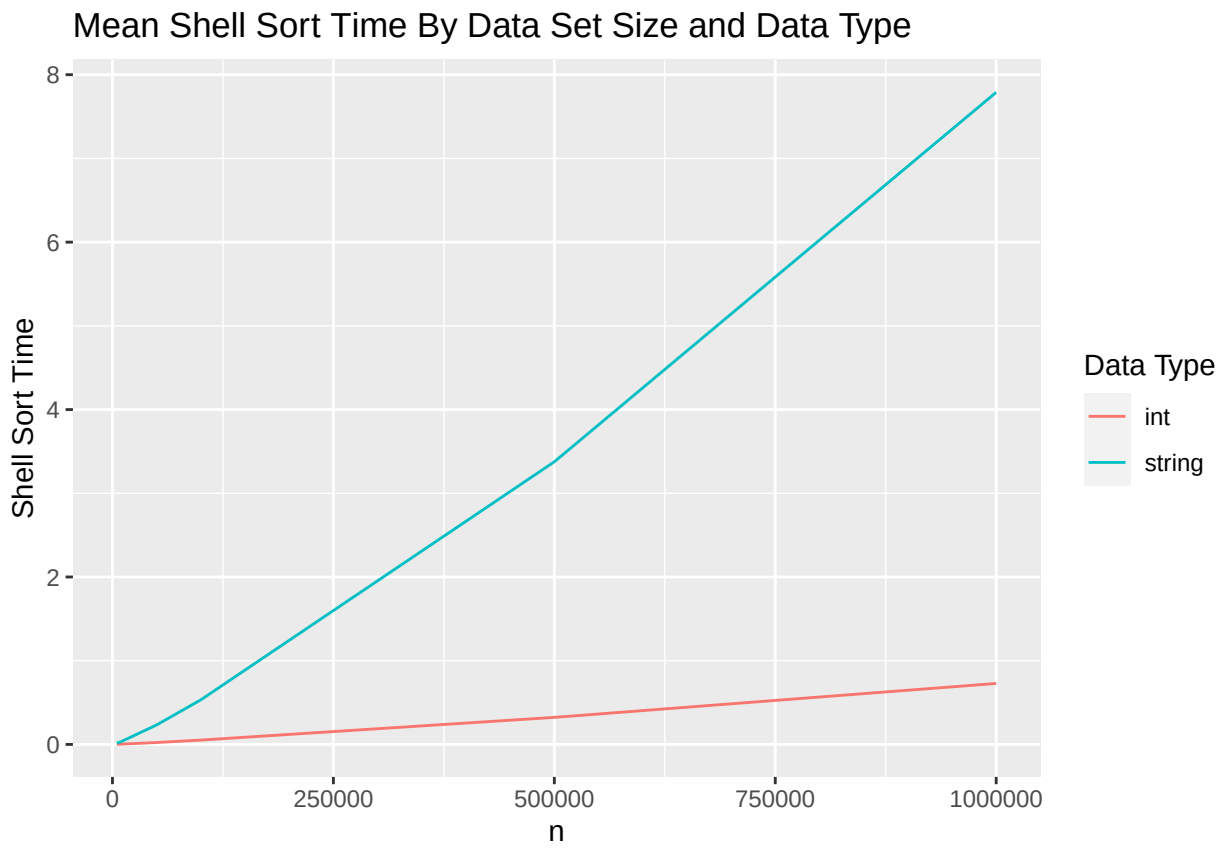

```
mergeInts = subset(mergeTimes, var_type == "int")
ggplot(mergeInts, aes(x = size, y = merge_time, color = format)) +
  geom_line() +
  labs(title = "Merge Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Merg
  guides(color = guide_legend(title = "File Format"))
```




```
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

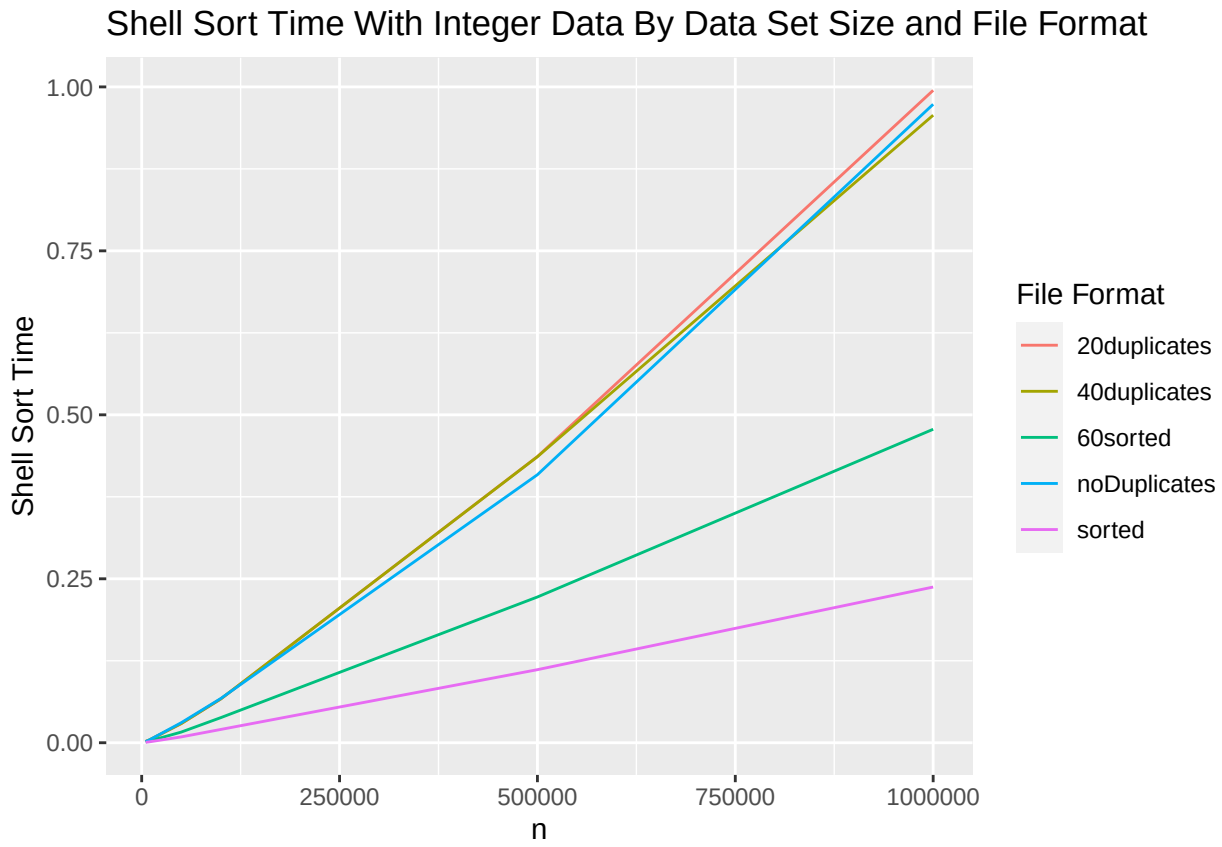
```
shellTimes = aggregate(shell_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
shellTimes2 = aggregate(shell_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(shellTimes2, aes(x = size, y = shell_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Shell Sort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

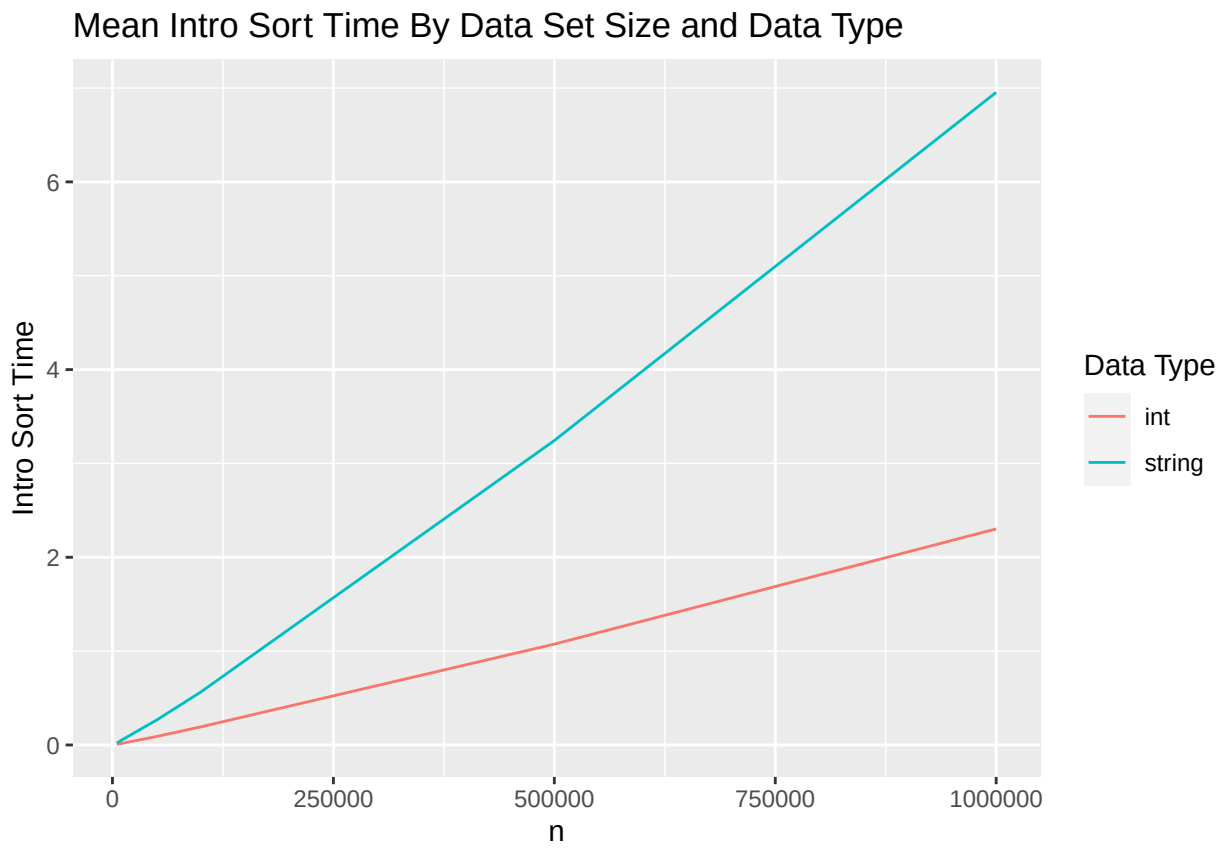

```
shellInts = subset(shellTimes, var_type == "int")
ggplot(shellInts, aes(x = size, y = shell_time, color = format)) +
  geom_line() +
  labs(title = "Shell Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Shell
  guides(color = guide_legend(title = "File Format"))
```




```
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"))
```

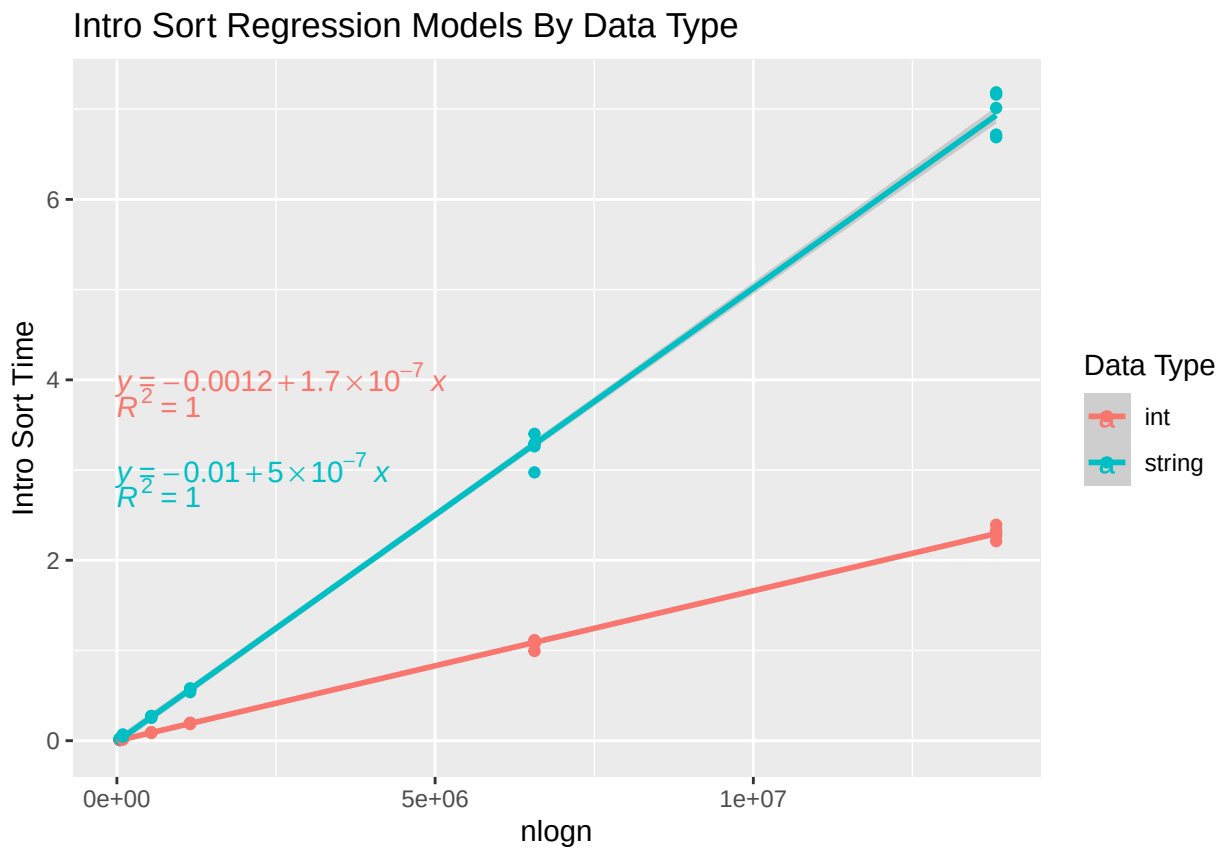
Intro Sort

```
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"))
```



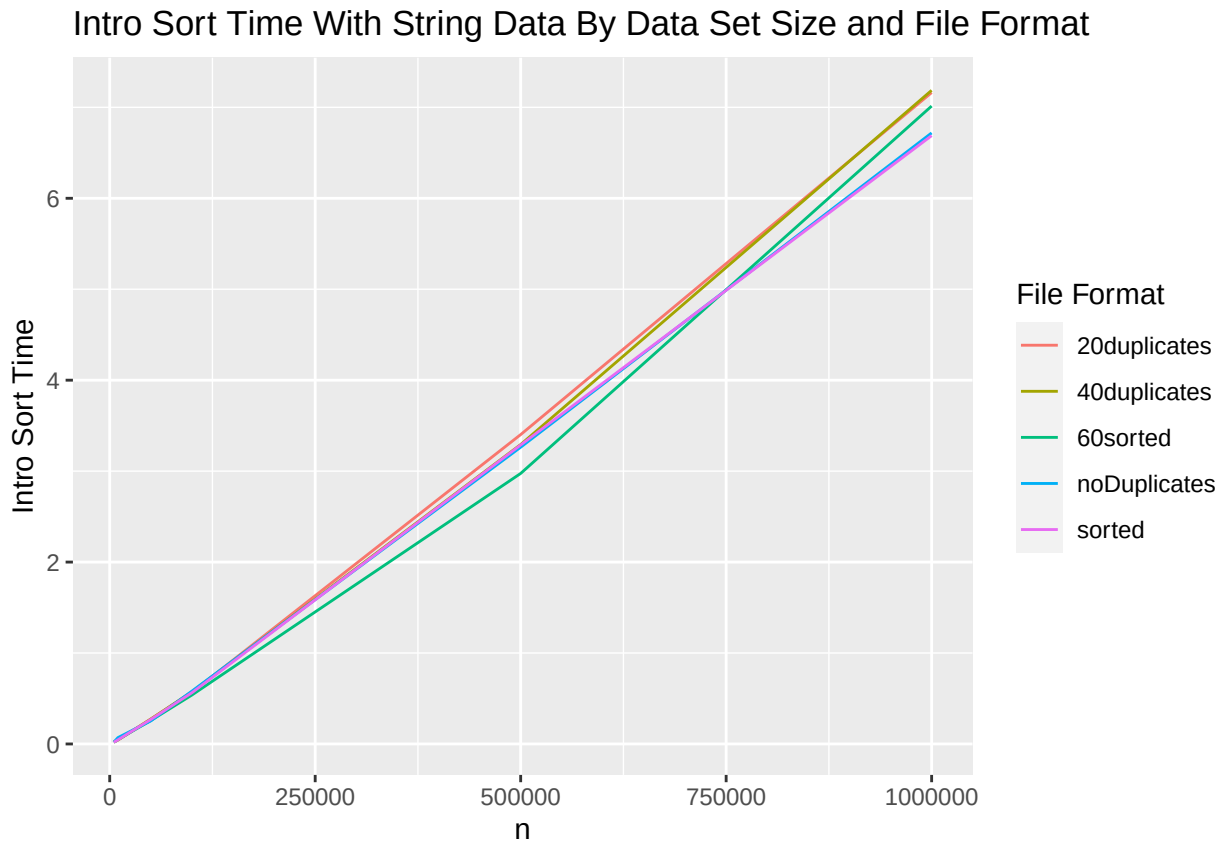

```
ggplot(introTimes, aes(x = nlogn, y = intro_time, color = var_type)) +
  labs(title = "Intro Sort Regression Models By Data Type", x = "nlogn", y = "Intro Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

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



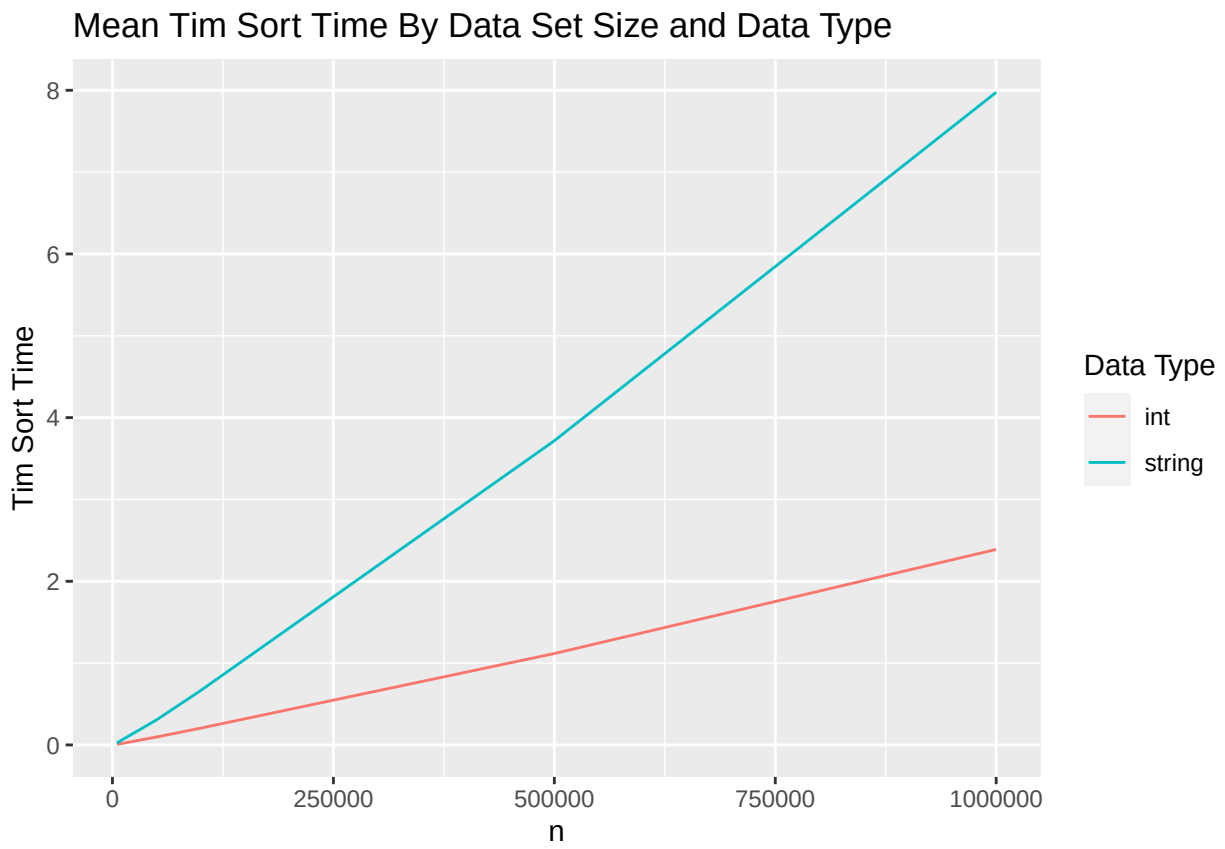

```
introInts = subset(introTimes, var_type == "int")
ggplot(introInts, aes(x = size, y = intro_time, color = format)) +
  geom_line() +
  labs(title = "Intro Sort Time With Integer Data By Data Set Size and File Format", x = "n", y = "Intro
  guides(color = guide_legend(title = "File Format"))
```


```
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

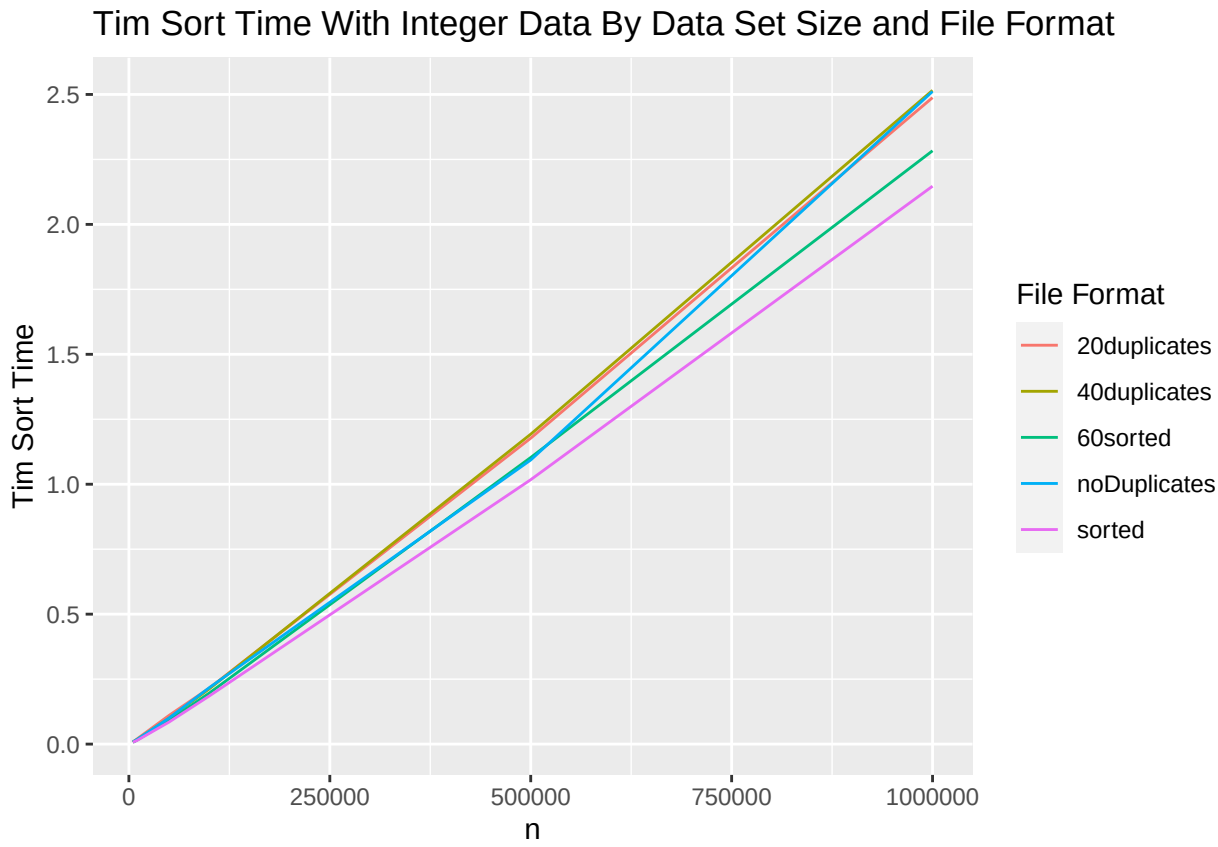
```
timTimes = aggregate(tim_time ~ var_type + size + nlogn + format, data = data, FUN = mean)
timTimes2 = aggregate(tim_time ~ var_type + size + nlogn, data = data, FUN = mean)
ggplot(timTimes2, aes(x = size, y = tim_time, color = var_type)) +
  geom_line() +
  labs(title = "Mean Tim Sort Time By Data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



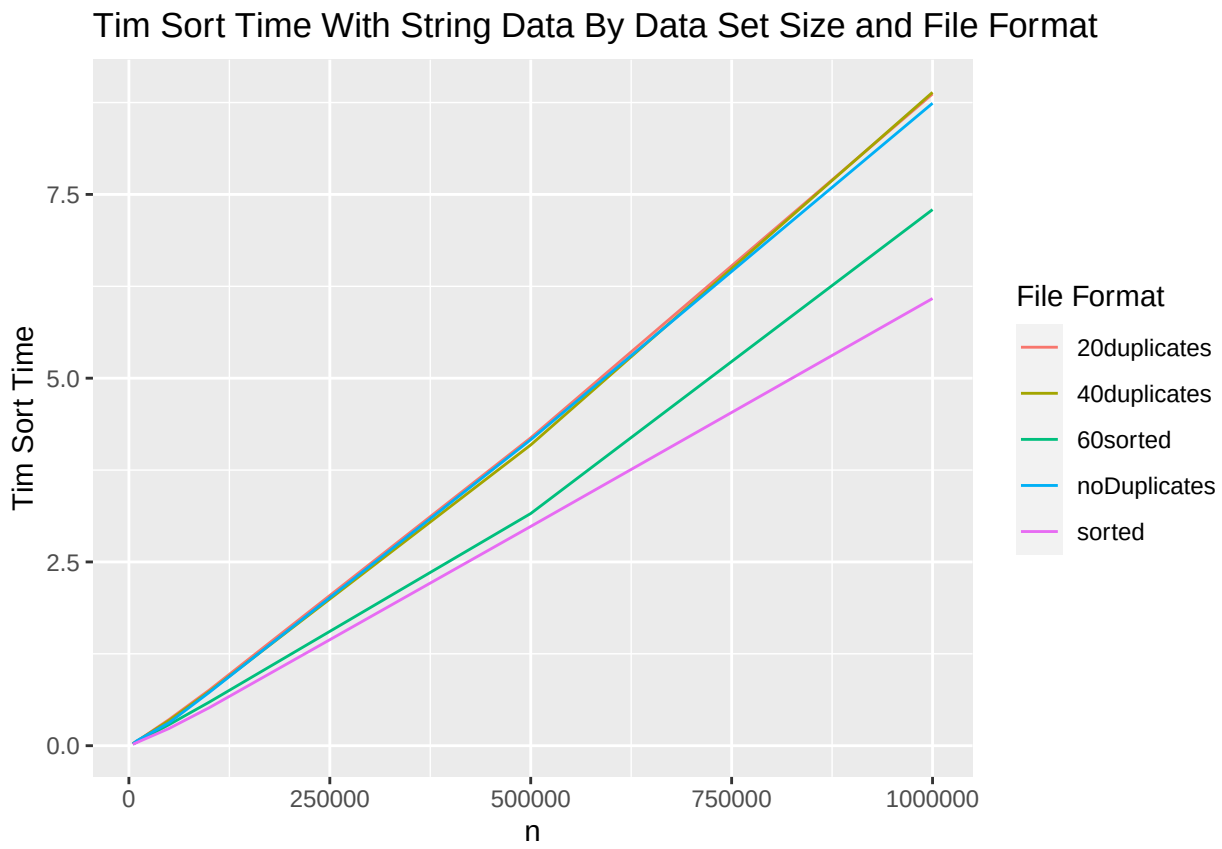

```
ggplot(timTimes, aes(x = nlogn, y = tim_time, color = var_type)) +
  labs(title = "Tim Sort Regression Models By Data Type", x = "nlogn", y = "Tim Sort Time") +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=0, label.y=c(4, 3)) +
  stat_cor(aes(label=..rr.label..),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

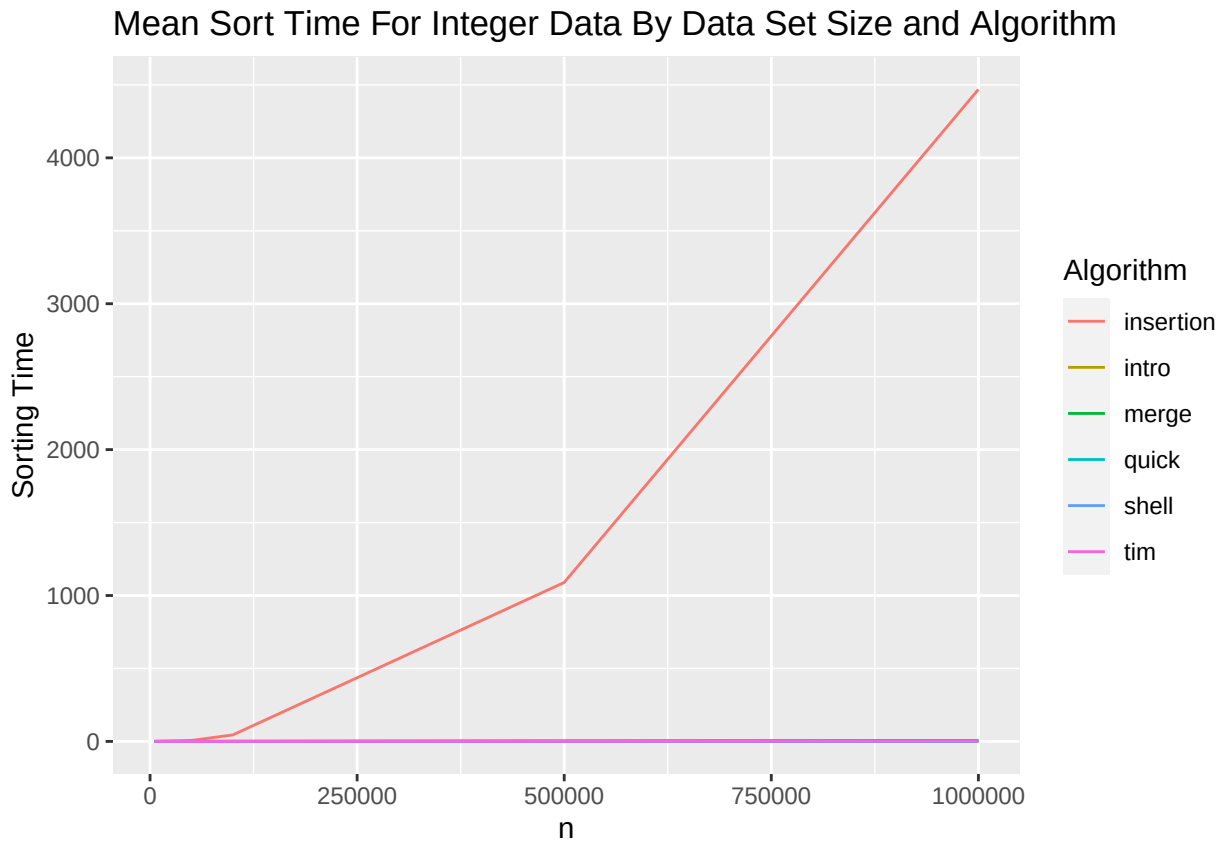


Algorithm Comparison

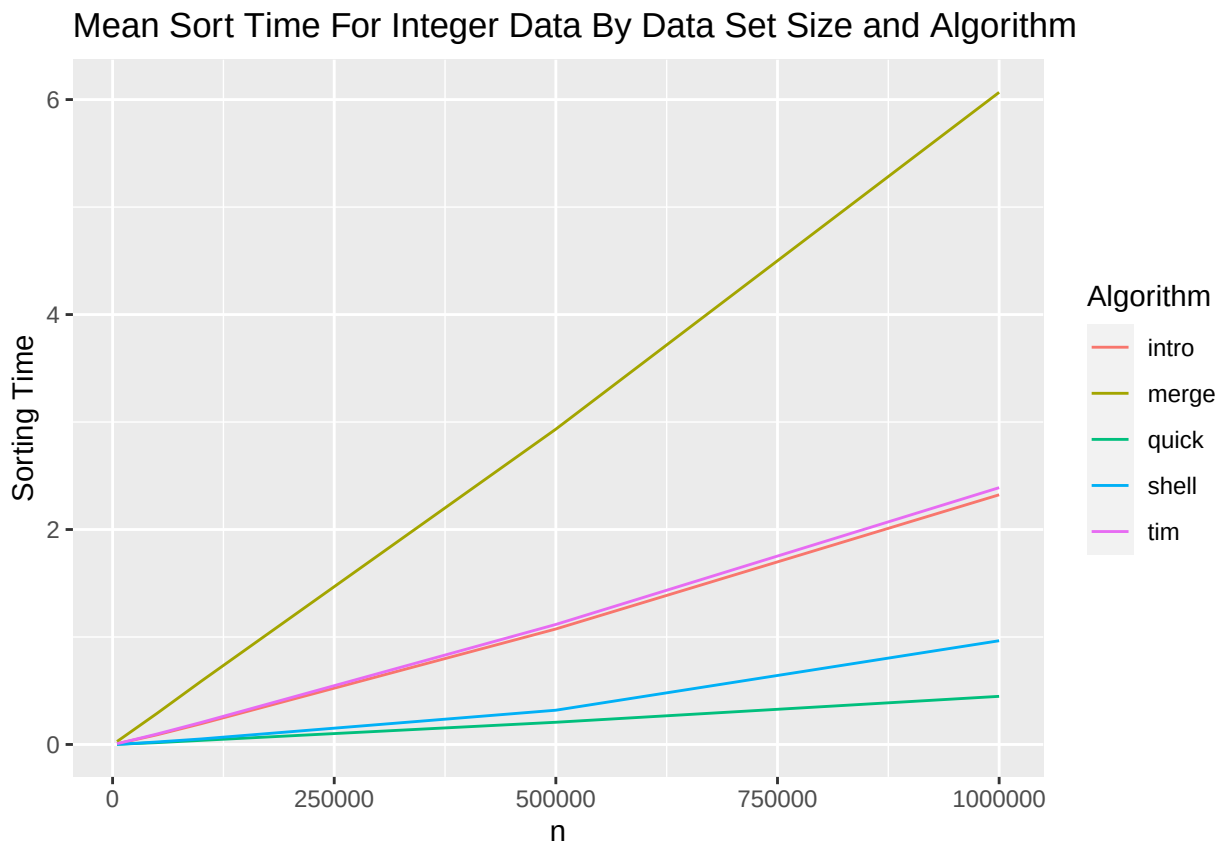
```
data2 = matrix(ncol = 5, nrow = 360)
for (i in 1:6) {
  for (j in 1:60) {
    data2[i * j, 1] = data[j, 1]
    data2[i * j, 2] = data[j, 2]
    data2[i * j, 3] = data[j, 3]
    data2[i * j, 4] = data[j, 3 + i]
    if (i == 1) {
      data2[i * j, 5] = "insertion"
    } else if (i == 2) {
      data2[i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

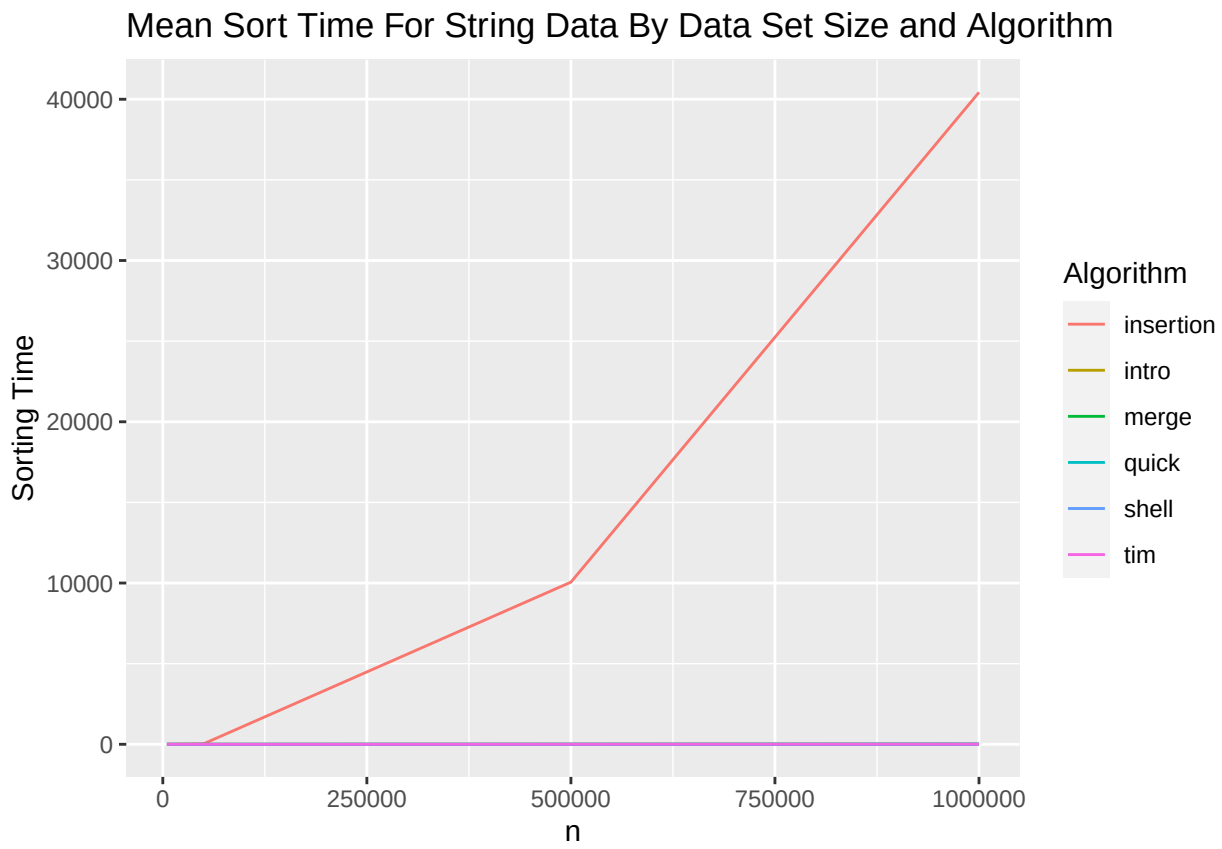
```
integerData = subset(data2, var_type == "int")
integerTimes = aggregate(time ~ algorithm + size, data = integerData, FUN = mean)
ggplot(integerTimes, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x =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



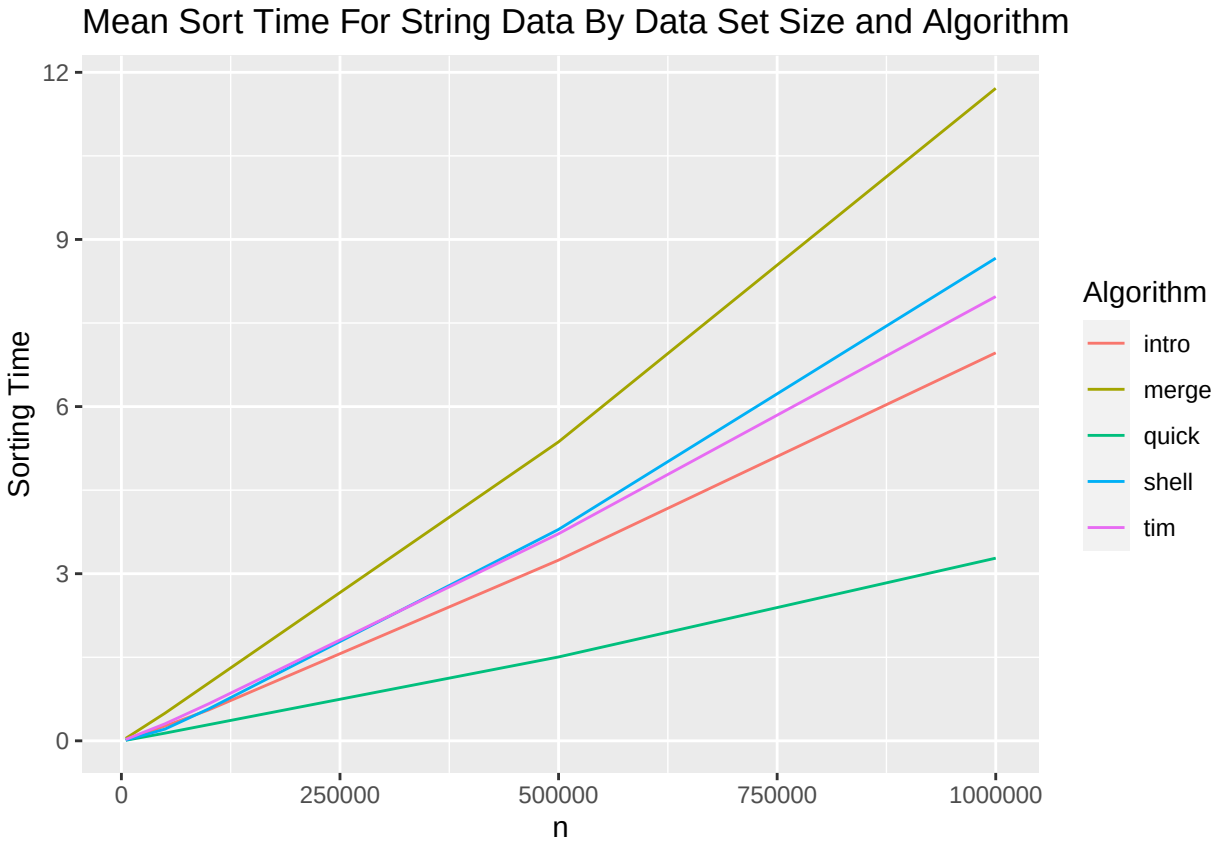

```
integerTimes2 = subset(integerTimes, algorithm != "insertion")
ggplot(integerTimes2, aes(x = size, y = time, color = algorithm)) +
  geom_line() +
  labs(title = "Mean Sort Time For Integer Data By Data Set Size and Algorithm", x = "n", y = "Sorting T
  guides(color = guide_legend(title = "Algorithm"))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




```
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
12
9
6
3
0
0
250000
500000
750000
1000000
n
Algorithm
intro
merge
quick
shell
tim