Regression with R

Data

For a certain algorithm the time to calculate the results is measured for different input sizes and recorded in a text file.

Read this data into R using the read.table function.

```
# Read the data
data = read.csv("quickSort.csv", sep=',', header=FALSE)
colnames(data) = c("Size","Duration")
data
```

```
##         Size    Duration
## 1         10      0.0009
## 2         20      0.0015
## 3         30      0.0027
## 4         40      0.0024
## 5         50      0.0033
## 6         60      0.0042
## 7         70      0.0042
## 8         80      0.0058
## 9         90      0.0076
## 10       100      0.0090
## 11       200      0.0151
## 12       300      0.0226
## 13       400      0.0317
## 14       500      0.0417
## 15       600      0.0604
## 16       700      0.0624
## 17       800      0.0727
## 18       900      0.1096
## 19      1000      0.1120
## 20      2000      0.2527
## 21      3000      0.3985
## 22      4000      0.4310
## 23      5000      0.5974
## 24      6000      0.8643
## 25      7000      0.9300
## 26      8000      0.9877
## 27      9000      1.2186
## 28     10000      1.3415
## 29     20000      3.5170
## 30     30000      6.4963
## 31     40000      9.0060
## 32     50000     12.9166
## 33     60000     18.2126
## 34     70000     22.1514
## 35     80000     27.6100
## 36     90000     33.0320
## 37    100000     39.4426
## 38    200000    136.9890
```

```
## 39   300000    286.5380
## 40   400000    507.6420
## 41   500000    759.4430
## 42   600000   1082.9900
## 43   700000   1481.0300
## 44   800000   1894.6600
## 45   900000   2391.5300
## 46  1000000   2926.5000
## 47  2000000  11629.0000
## 48  3000000  25835.1000
## 49  4000000  45585.9000
## 50  5000000  71401.8000
## 51  6000000 105862.0000
## 52  7000000 141372.0000
## 53  8000000 191930.0000
## 54  9000000 234185.0000
## 55 10000000 289828.0000
```

Data analysis: log10 transformation

Let's see how well the data fits a model after a log10 transform.

```
logn = log10(data$Size)
mlogn = lm(data$Duration ~ logn)
summary(mlogn)
```

```
##
## Call:
## lm(formula = data$Duration ~ logn)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -50331 -33097 -10372  18689 218686
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -54048      17645  -3.063  0.00344 **
## logn           17884       3898   4.588 2.78e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50880 on 53 degrees of freedom
## Multiple R-squared:  0.2842, Adjusted R-squared:  0.2707
## F-statistic: 21.05 on 1 and 53 DF,  p-value: 2.782e-05
```

Data analysis: square transformation

Let's see how well the data fits a model after a n^2 transform.

```
nsq = data$Size^2
mnsq = lm(data$Duration ~ nsq)
summary(mnsq)
```

```
##
## Call:
## lm(formula = data$Duration ~ nsq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1689.3     4.7     5.1    13.2  5561.2
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.735e+00  1.264e+02  -0.037     0.97
## nsq          2.912e-09  5.891e-12 494.290   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 885.7 on 53 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 2.443e+05 on 1 and 53 DF,  p-value: < 2.2e-16
```

Data analysis: n*log10 transformation

Let's see how well the data fits a model after a n*log10(n) transform.

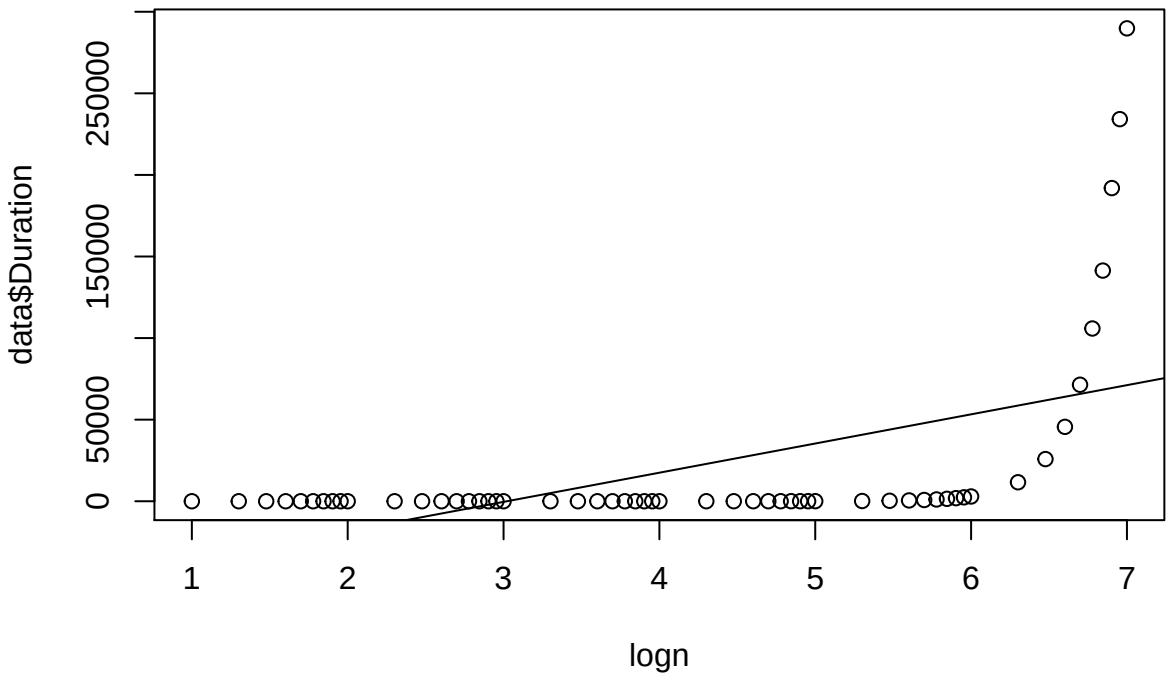
```
nlogn = data$Size * log10(data$Duration)
mnlogn = lm(data$Duration ~ nlogn)
summary(mnlogn)
```

```
##
## Call:
## lm(formula = data$Duration ~ nlogn)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35136   1167   3590   3592  46295
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.590e+03  1.808e+03  -1.986   0.0522 .
## nlogn        4.524e-03  1.306e-04  34.635   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12370 on 53 degrees of freedom
## Multiple R-squared:  0.9577, Adjusted R-squared:  0.9569
## F-statistic:  1200 on 1 and 53 DF,  p-value: < 2.2e-16
```

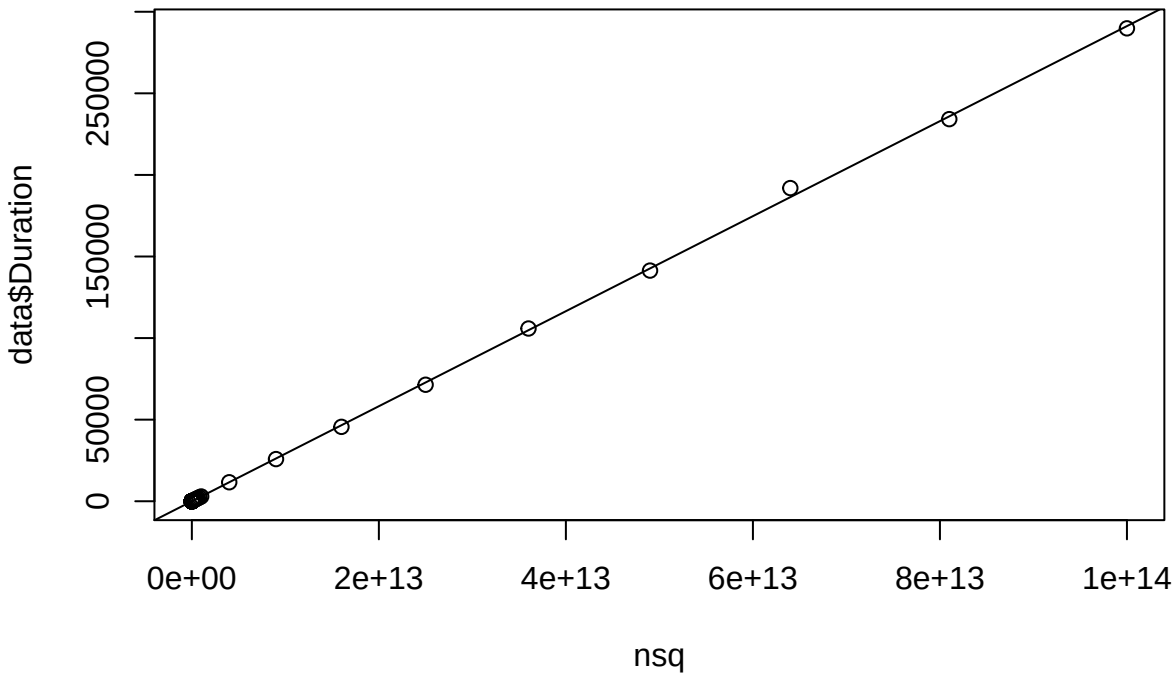
Visual analysis

Create graphs for each combination. Plot the regression line.

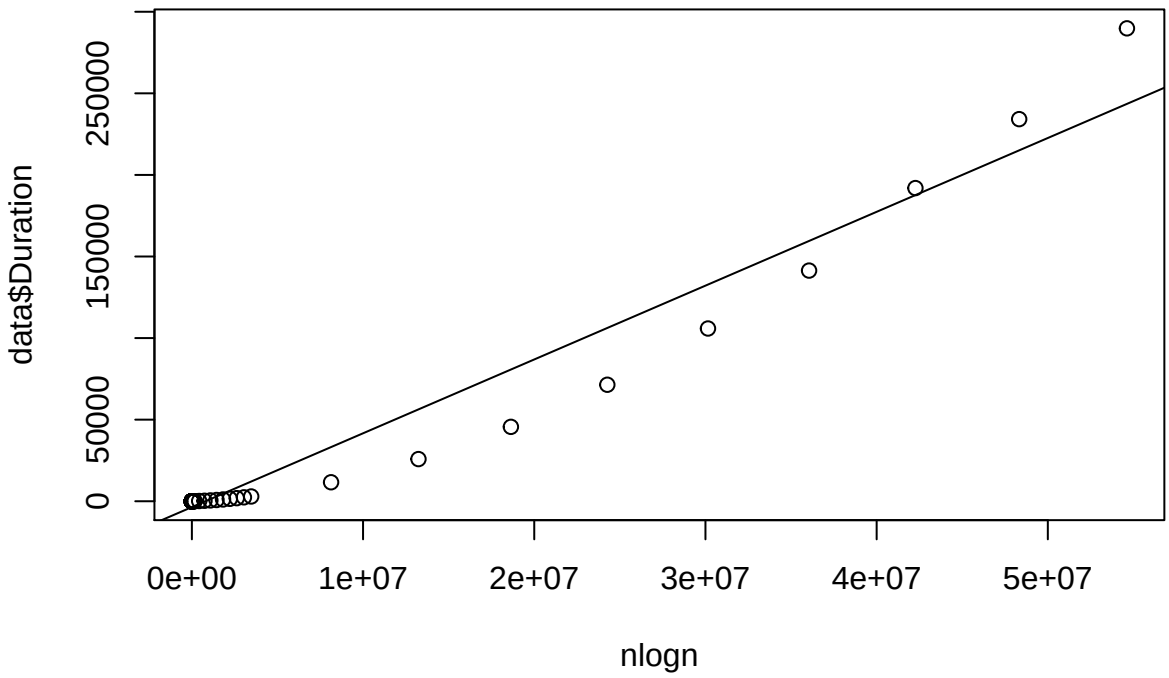
```
plot(logn,data$Duration)
abline(mlogn)
```




```
plot(nsq,data$Duration)
abline(mnsq)
```




```
plot(nlogn,data$Duration)
abline(mnlogn)
```



Conclusion

The R-square (R^2) value indicates the fit. Find the model where R^2 is the largest. The closer it is to 1, the better the fit. In this example nsq has the best fit. Therefore we can conclude that the algorithm likely has an order of O(n^2). Altough this is the same as wikipedia says, wikipedia also says that this is the worst case performance for the quicksort algorithm this could be because pivot is always the highest or lowest number in the array but this is very unlikely in our code.