Untitled

GrJa23

November 23, 2023

Analysis Over the efficiency of a multi-threaded implementation of the QuickSort algorithm on multi-core machines

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

df <- read.csv("measurements_03 47.csv", header = T)
df$Type <- gsub("\\s+", "", df$Type)
head(df)
```

```
##   Size       Type     Time
## 1  100 Sequential 0.000010
## 2  100   Parallel 0.004024
## 3  100   Built-in 0.000013
## 4  100 Sequential 0.000010
## 5  100   Parallel 0.004448
## 6  100   Built-in 0.000014
```

```
summary(df)
```

```
##       Size             Type                Time
##  Min.   :    100   Length:75          Min.   :0.000009
##  1st Qu.:   1000   Class :character   1st Qu.:0.000210
##  Median :  10000   Mode  :character   Median :0.016149
##  Mean   : 222220                      Mean   :0.051255
##  3rd Qu.: 100000                      3rd Qu.:0.043877
##  Max.   :1000000                      Max.   :0.242869
```

```
print(df)
```

```
##       Size       Type     Time
## 1      100 Sequential 0.000010
```

```
## 2      100   Parallel 0.004024
## 3      100   Built-in 0.000013
## 4      100 Sequential 0.000010
## 5      100   Parallel 0.004448
## 6      100   Built-in 0.000014
## 7      100 Sequential 0.000009
## 8      100   Parallel 0.003384
## 9      100   Built-in 0.000013
## 10     100 Sequential 0.000010
## 11     100   Parallel 0.003738
## 12     100   Built-in 0.000012
## 13     100 Sequential 0.000010
## 14     100   Parallel 0.003133
## 15     100   Built-in 0.000011
## 16    1000 Sequential 0.000128
## 17    1000   Parallel 0.020407
## 18    1000   Built-in 0.000209
## 19    1000 Sequential 0.000126
## 20    1000   Parallel 0.022003
## 21    1000   Built-in 0.000201
## 22    1000 Sequential 0.000128
## 23    1000   Parallel 0.016149
## 24    1000   Built-in 0.000210
## 25    1000 Sequential 0.000128
## 26    1000   Parallel 0.014594
## 27    1000   Built-in 0.000209
## 28    1000 Sequential 0.000129
## 29    1000   Parallel 0.014905
## 30    1000   Built-in 0.000210
## 31   10000 Sequential 0.001774
## 32   10000   Parallel 0.018943
## 33   10000   Built-in 0.001720
## 34   10000 Sequential 0.001698
## 35   10000   Parallel 0.016226
## 36   10000   Built-in 0.001733
## 37   10000 Sequential 0.001652
## 38   10000   Parallel 0.017348
## 39   10000   Built-in 0.001702
## 40   10000 Sequential 0.001680
## 41   10000   Parallel 0.017302
## 42   10000   Built-in 0.001726
## 43   10000 Sequential 0.001675
## 44   10000   Parallel 0.017386
## 45   10000   Built-in 0.001716
## 46  100000 Sequential 0.020040
## 47  100000   Parallel 0.050548
## 48  100000   Built-in 0.020300
## 49  100000 Sequential 0.020004
## 50  100000   Parallel 0.043119
## 51  100000   Built-in 0.020504
## 52  100000 Sequential 0.019763
## 53  100000   Parallel 0.050735
## 54  100000   Built-in 0.020439
## 55  100000 Sequential 0.019913
```

```
## 56  100000   Parallel 0.049806
## 57  100000   Built-in 0.020541
## 58  100000 Sequential 0.019726
## 59  100000   Parallel 0.044636
## 60  100000   Built-in 0.020252
## 61 1000000 Sequential 0.230648
## 62 1000000   Parallel 0.162221
## 63 1000000   Built-in 0.242869
## 64 1000000 Sequential 0.235778
## 65 1000000   Parallel 0.162137
## 66 1000000   Built-in 0.241607
## 67 1000000 Sequential 0.238383
## 68 1000000   Parallel 0.163279
## 69 1000000   Built-in 0.242786
## 70 1000000 Sequential 0.232921
## 71 1000000   Parallel 0.170237
## 72 1000000   Built-in 0.241583
## 73 1000000 Sequential 0.230096
## 74 1000000   Parallel 0.153896
## 75 1000000   Built-in 0.242492
```

Including Plots

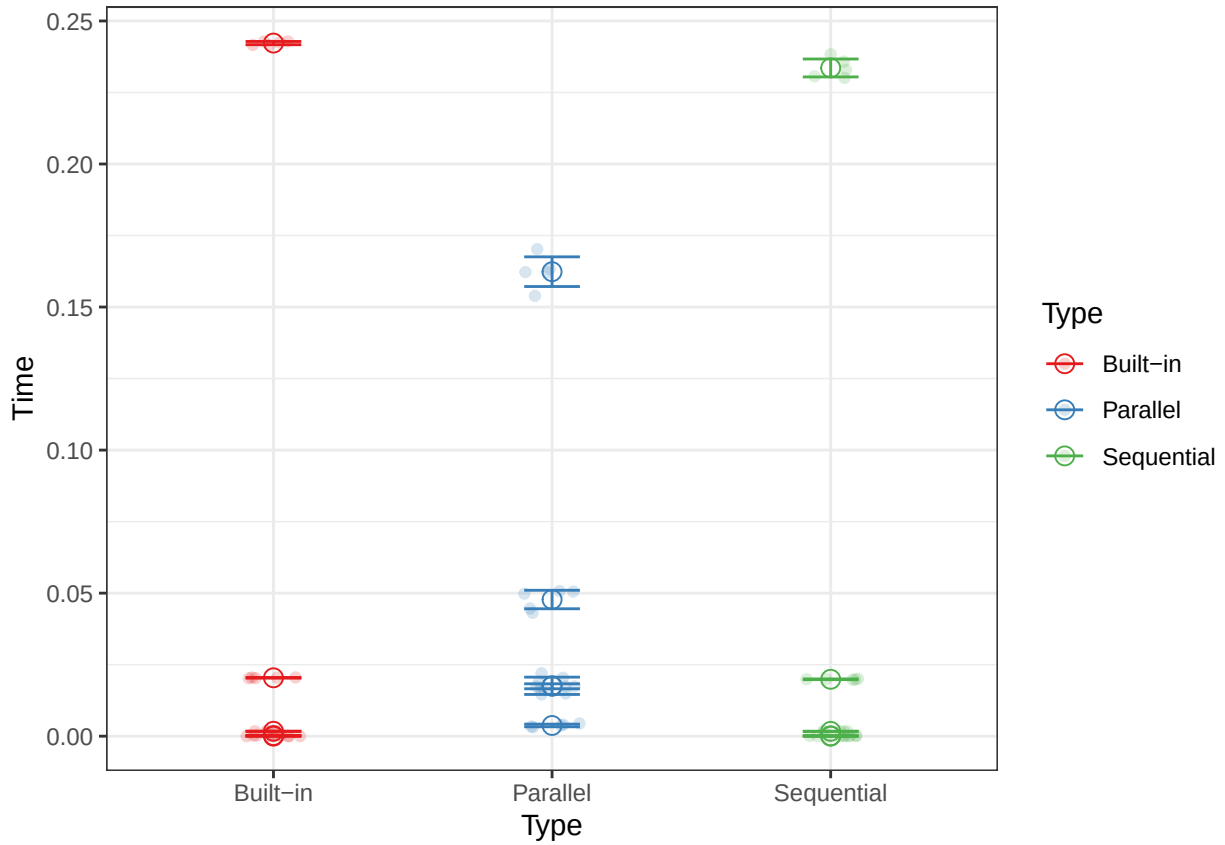
You can also embed plots, for example:

```
dfsum <- group_by(df, Size, Type) %>%
  summarise(num = n(), mean = mean(Time), sd = sd(Time), se = 2*sd/sqrt(num),.groups = 'drop')
```

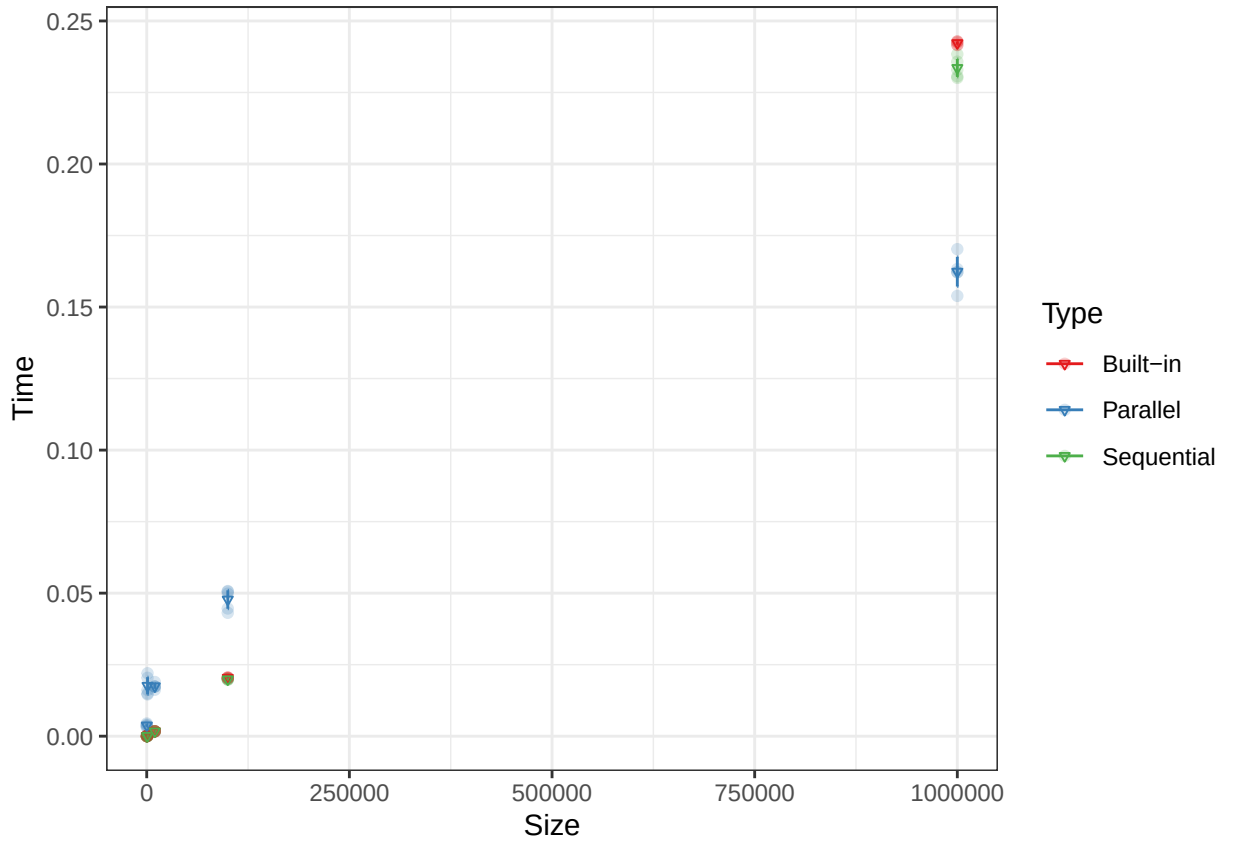
```
print(dfsum)
```

```
## # A tibble: 15 x 6
##       Size Type         num      mean          sd          se
##      <int> <chr>      <int>     <dbl>       <dbl>       <dbl>
##  1     100 Built-in       5 0.0000126 0.00000114  0.00000102
##  2     100 Parallel       5 0.00375   0.000519    0.000464
##  3     100 Sequential     5 0.0000098 0.000000447 0.000000400
##  4    1000 Built-in       5 0.000208  0.00000383  0.00000343
##  5    1000 Parallel       5 0.0176    0.00338     0.00302
##  6    1000 Sequential     5 0.000128  0.00000110  0.000000980
##  7   10000 Built-in       5 0.00172   0.0000117   0.0000104
##  8   10000 Parallel       5 0.0174    0.000970    0.000868
##  9   10000 Sequential     5 0.00170   0.0000467   0.0000418
## 10  100000 Built-in       5 0.0204    0.000126    0.000113
## 11  100000 Parallel       5 0.0478    0.00361     0.00323
## 12  100000 Sequential     5 0.0199    0.000141    0.000126
## 13 1000000 Built-in       5 0.242     0.000630    0.000563
## 14 1000000 Parallel       5 0.162     0.00580     0.00519
## 15 1000000 Sequential     5 0.234     0.00350     0.00313
```

```
ggplot(df,aes(x = Type,y = Time,color = Type)) +
  scale_color_brewer(palette="Set1") + theme_bw() +
  geom_jitter(alpha=.2,position = position_jitter(width = .1)) +
  geom_errorbar(data=dfsum,width=.2, aes(y=mean,ymin=mean-se,ymax=mean+se)) +
  geom_point(data=dfsum,shape=21, size=3, aes(y=mean,color=Type))
```

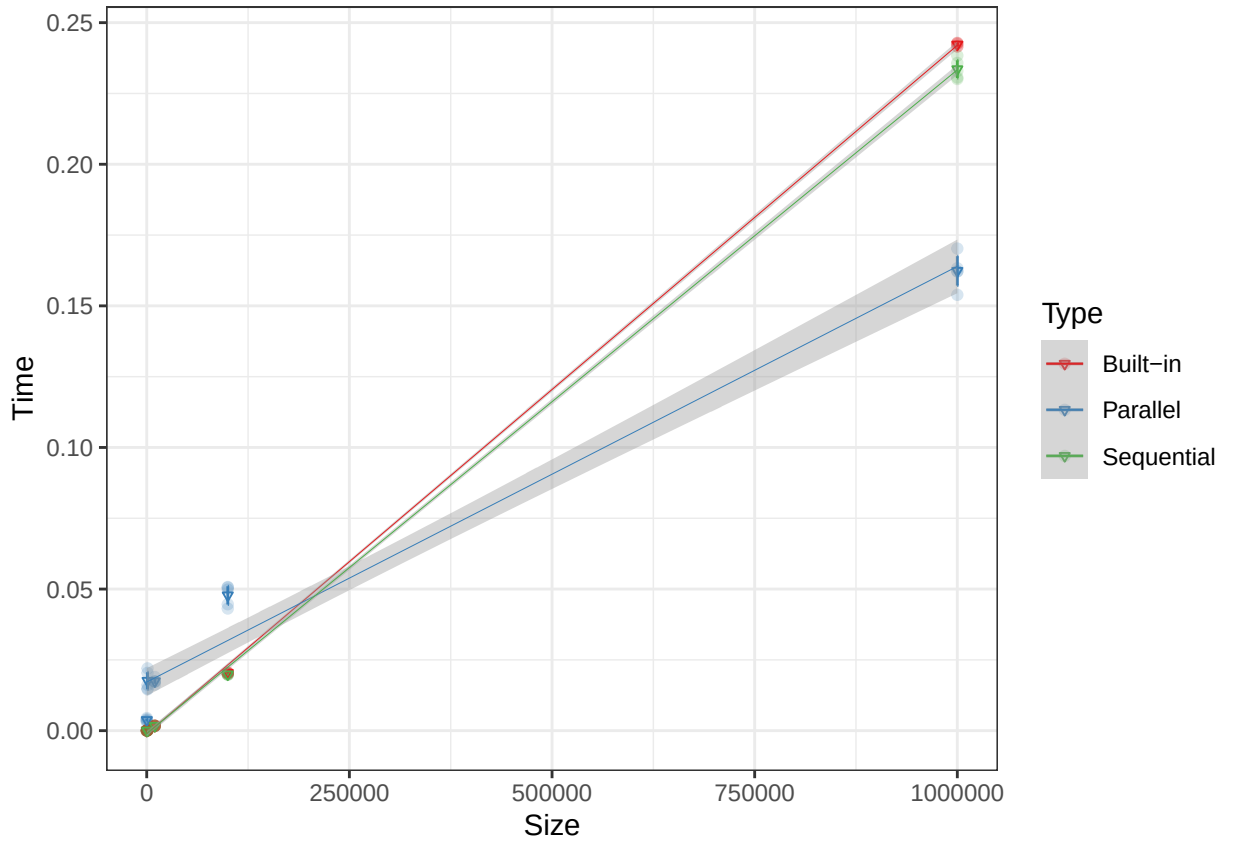



```
ggplot(df,aes(x=Size,y=Time,color=Type)) +
  scale_color_brewer(palette="Set1") + theme_bw() +
  geom_jitter(alpha=.2,position=position_jitter(width = 0.1)) +
  geom_errorbar(data=dfsum,width=0.1, aes(y=mean,ymin=mean-se,ymax=mean+se)) +
  geom_point(data=dfsum,shape=25, size=1, aes(y=mean,color=Type))
```




```
ggplot(df,aes(x=Size,y=Time,color=Type)) +
  scale_color_brewer(palette="Set1") + theme_bw() +
  geom_jitter(alpha=.2,position = position_jitter(width = 0.1)) +
  geom_errorbar(data=dfsum,width=0.1, aes(y=mean,ymin=mean-se,ymax=mean+se)) +
  geom_point(data=dfsum,shape=25, size=1, aes(y=mean,color=Type))+
  geom_smooth(method="lm",linewidth=0.1)
```

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



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
dfforlm = df %>% filter(Type == "Parallel")
print(dfforlm)
```

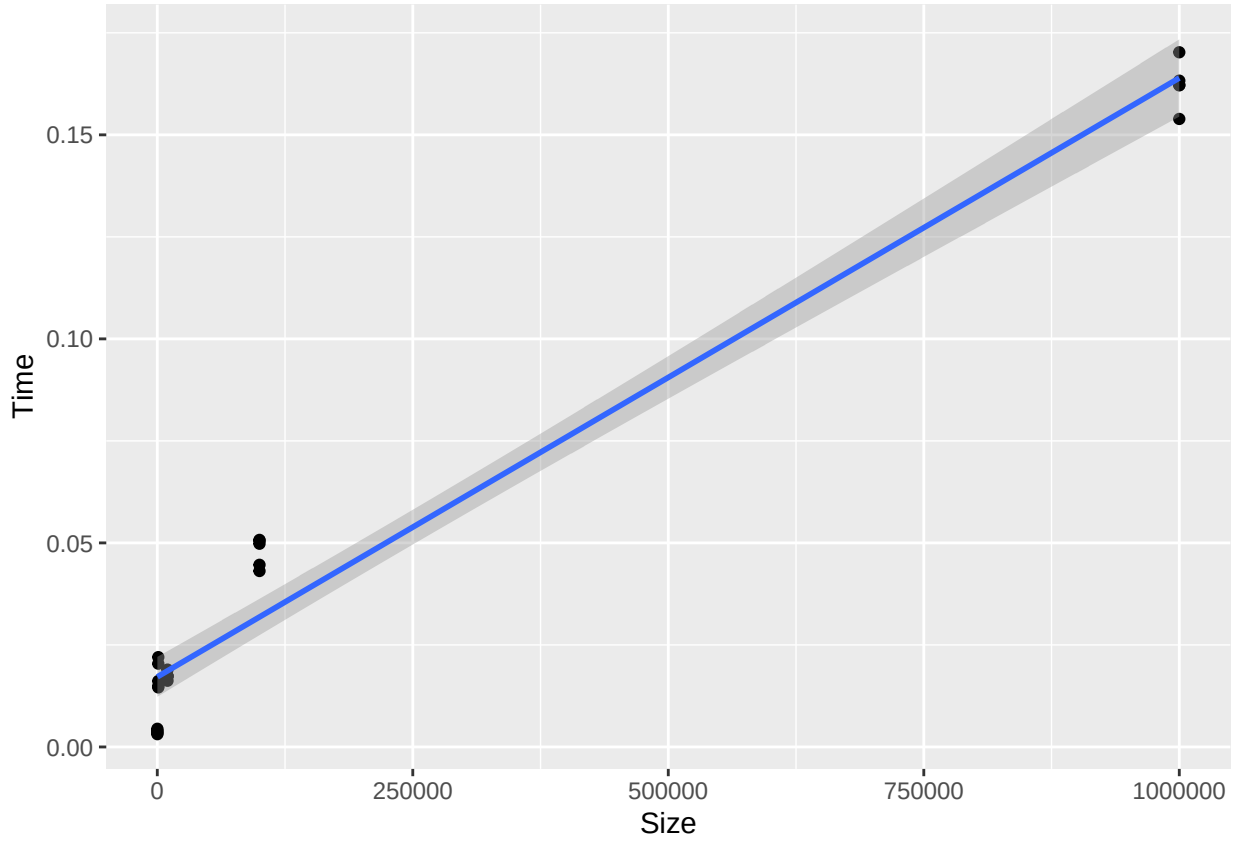
```
##       Size     Type     Time
## 1      100 Parallel 0.004024
## 2      100 Parallel 0.004448
## 3      100 Parallel 0.003384
## 4      100 Parallel 0.003738
## 5      100 Parallel 0.003133
## 6     1000 Parallel 0.020407
## 7     1000 Parallel 0.022003
## 8     1000 Parallel 0.016149
## 9     1000 Parallel 0.014594
## 10    1000 Parallel 0.014905
## 11   10000 Parallel 0.018943
## 12   10000 Parallel 0.016226
## 13   10000 Parallel 0.017348
## 14   10000 Parallel 0.017302
## 15   10000 Parallel 0.017386
## 16  100000 Parallel 0.050548
## 17  100000 Parallel 0.043119
## 18  100000 Parallel 0.050735
## 19  100000 Parallel 0.049806
## 20  100000 Parallel 0.044636
```

```
## 21 1000000 Parallel 0.162221
## 22 1000000 Parallel 0.162137
## 23 1000000 Parallel 0.163279
## 24 1000000 Parallel 0.170237
## 25 1000000 Parallel 0.153896
```

```
#reg <- lm(y ~ x,data = dfforlm)
reg <- lm(Time ~ Size,data = dfforlm)
summary(reg)
```

```
##
## Call:
## lm(formula = Time ~ Size, data = dfforlm)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.014052 -0.002723 -0.001290  0.004686  0.018888
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.717e-02  2.364e-03   7.263 2.16e-07 ***
## Size        1.468e-07  5.260e-09  27.904  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01027 on 23 degrees of freedom
## Multiple R-squared:  0.9713, Adjusted R-squared:  0.9701
## F-statistic: 778.6 on 1 and 23 DF,  p-value: < 2.2e-16
```

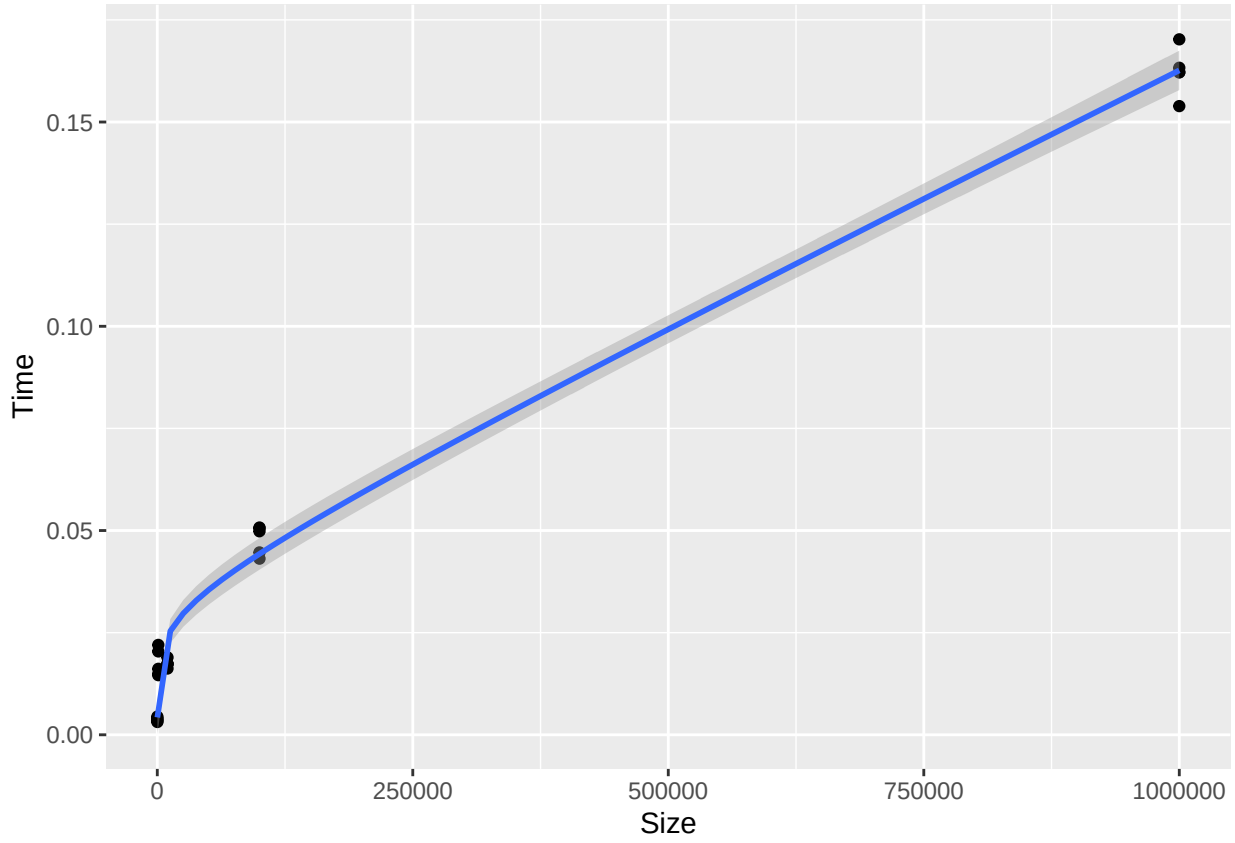
```
ggplot(dfforlm,aes(x = Size, y = Time)) + geom_point() + stat_smooth(method = "lm", formula = y ~ x,geo
```




```
reg2 <- lm(Time ~ log(Size)+Size^2,data = dfforlm)
summary(reg2)
```

```
##
## Call:
## lm(formula = Time ~ log(Size) + Size^2, data = dfforlm)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0087444 -0.0011620 -0.0002334  0.0024763  0.0083303
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.437e-02  4.022e-03  -3.572   0.0017 **
## log(Size)    4.042e-03  4.920e-04   8.214  3.8e-08 ***
## Size         1.212e-07  4.101e-09  29.546  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005209 on 22 degrees of freedom
## Multiple R-squared:  0.9929, Adjusted R-squared:  0.9923
## F-statistic:  1548 on 2 and 22 DF,  p-value: < 2.2e-16
```

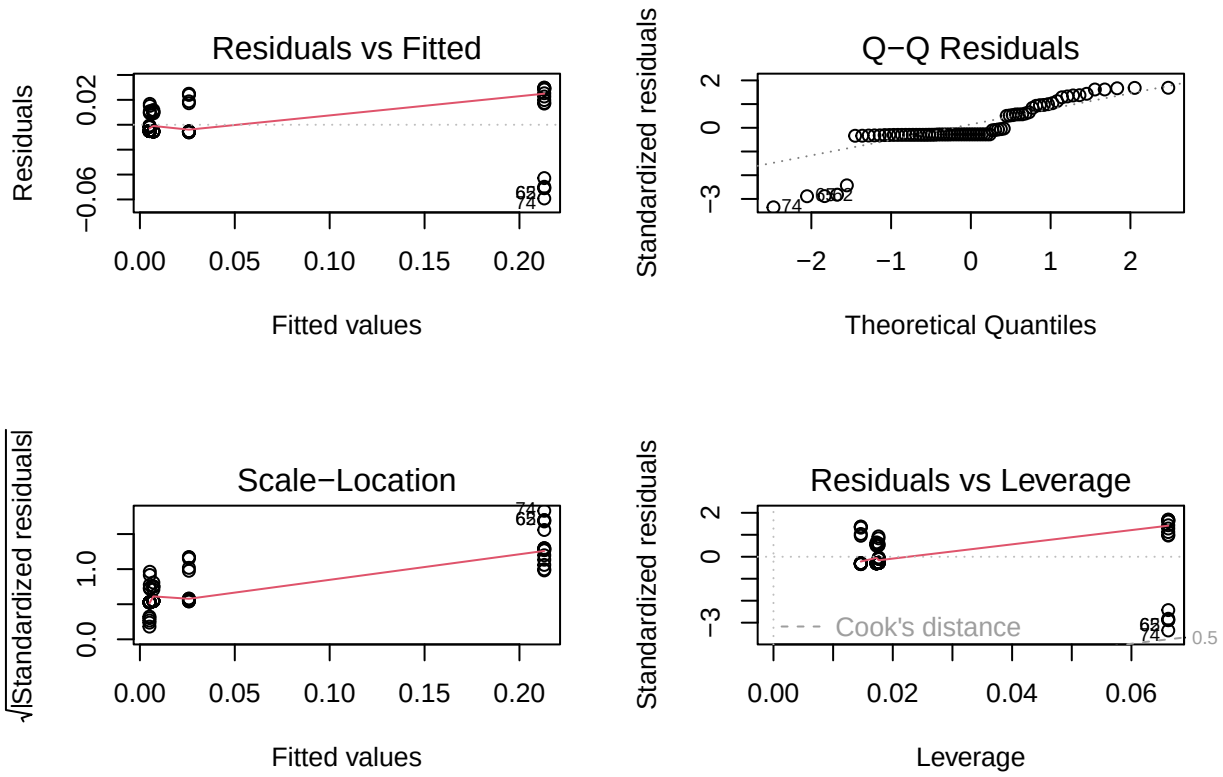
```
ggplot(dfforlm,aes(x = Size, y = Time)) + geom_point() + stat_smooth(method = "lm", formula = y ~ log(x
```



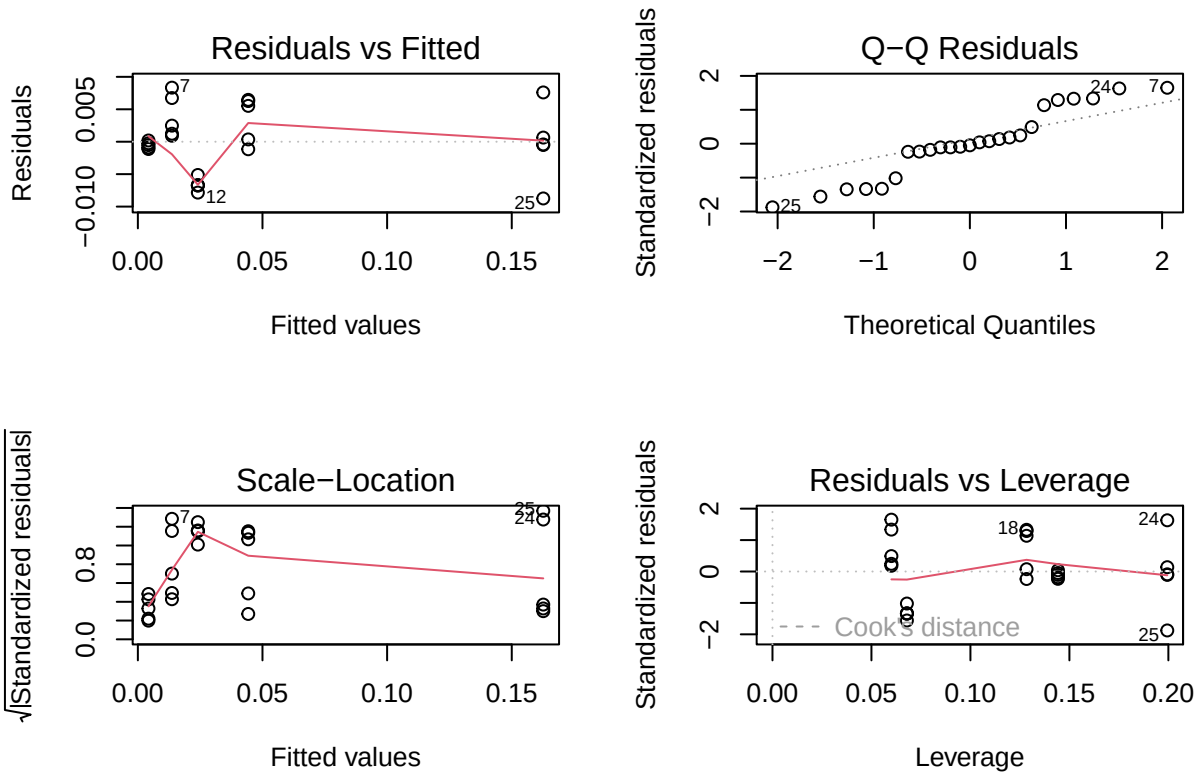

```
reg3 <- lm(Time ~ Size,data = df)
summary(reg3)
```

```
##
## Call:
## lm(formula = Time ~ Size, data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059185 -0.005382 -0.005027  0.010604  0.029788
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.020e-03  2.426e-03   2.069   0.0421 *
## Size        2.081e-07  5.398e-09  38.544   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01826 on 73 degrees of freedom
## Multiple R-squared:  0.9532, Adjusted R-squared:  0.9525
## F-statistic:  1486 on 1 and 73 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2));plot(reg3);par(mfrow=c(1,1))
```




```
par(mfrow=c(2,2));plot(reg2);par(mfrow=c(1,1))
```


Residuals vs Fitted
Residuals
Fitted values
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage