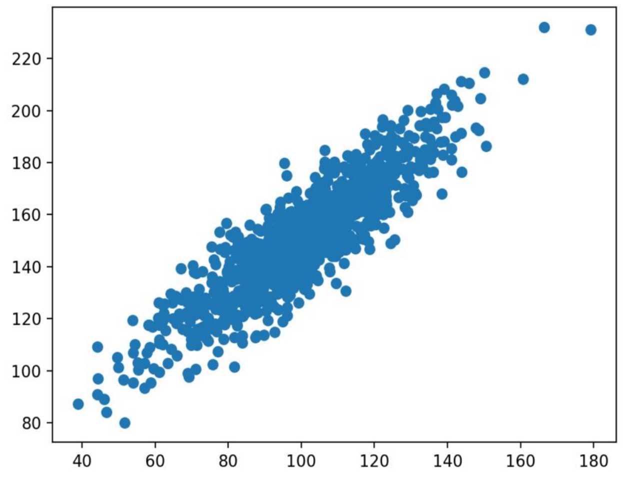
Scientific Methodology and Performance Evaluation

Subject : Latency and capacity estimation for a network connection for asymmetric measurements



Student : SADKI Abdelwahab

Student : Pr. Arnaud Legrand
Pr. Jean-Marc Vincent

Introduction:

The Goal of this project is to analyse the results from an experiment that has been done to study the performance of a given network. the theoretical model we have is: *T(S) = L + S/C.*

- T is the time required for sending a message.
- S is the size of the message.
- C is the capacity of the network/ Links.
- L is the latency.

In this model we are considering here is neglecting a lot of details.
With this work we want to find two parameters the Latency and the Capacity, to do that they used some random samples choosing different sizes and did some pings and at each time they calculate the time needed to do the transmission.
We have two different datasets we need to analyse and do the linear regression.

I. Uploading and Cleaning the Data.

~~~
```{r}
library(dplyr)
df = read.table('liglab2.log', sep=' ' , na.strings = ""  , header=F , fill = TRUE  )
```
~~~

- uploading the file 'liglab2.log' from the local machine make sure that the file is in the work placement of your R project.

~~~
```{r}
df = df %>% select(V1, V2, V9)
```
~~~

- Selecting only the columns we need for our Analyse.

~~~
```{r}
colnames(df)=c('date' ,'size' , 'time' )
```
~~~

- Giving the columns significant names to make the analyse easy.

~~~
```{r}
library(tidyr)
line_NA  = apply(df , 1 , function(x) any(is.na(x)))
df = df %>% drop_na()
```
~~~

- Removing all the lines with non available data.

~~~
```{r}
convertTime = function(time)
  gsub("[^0-9.]", "", time)

df$time = as.numeric(sapply(df$time , convertTime))
```
~~~

- Converting the time to a float.

~~~
```{r}
convertdate = function(date)
  gsub("[^0-9.]", "", date)


df$date = as.numeric(sapply(df$date , convertdate))
```
~~~

- converting the strings in the date column to significant date.

```{r}
head(df)
plot(time~size,df)
```

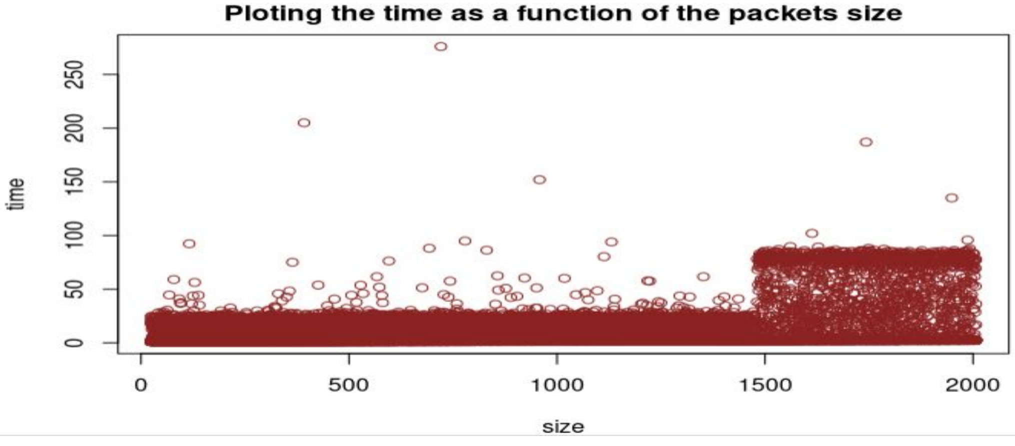
	date	size	time
	<dbl>	<int>	<dbl>
1	1421761682	665	22.50
2	1421761682	1373	21.20
3	1421761683	262	21.20
4	1421761683	1107	23.30
5	1421761683	1128	1.41
6	1421761683	489	21.90

6 rows

- Here we can see that we have the data with the correct format for the date and the correct time for the size and the time.

II. Plotting the Total Data

```
plot(time~size,df, main="Ploting the time as a function of the packets size", col="brown4")
```

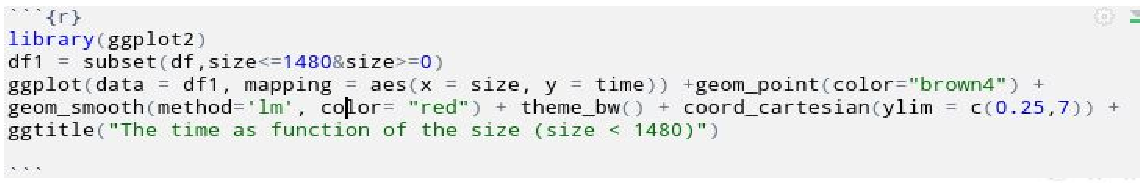


The figure above shows the time as a function of the size, from now we can see that we have two patterns. The behavior of the response time change differently when the size exceed 1500. This phenomena can be explained theoretically with the concept of fragmentation which means that when the packet size is greater than a given threshold this one will be divided and sent using more packets. Therefore we will have a big Latency.

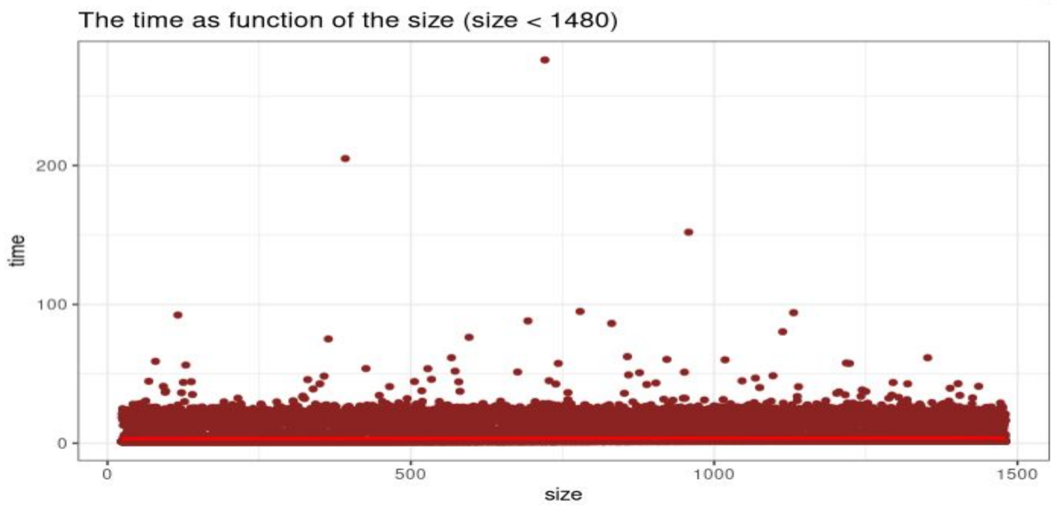
For this reason we need to split our dataset on two frames from 1480 bytes.

NB: In the markdown file i did the linear regression it was too bad. but i wanted to have something to compare the next models with maybe it will give something useful for understanding the problematic better.

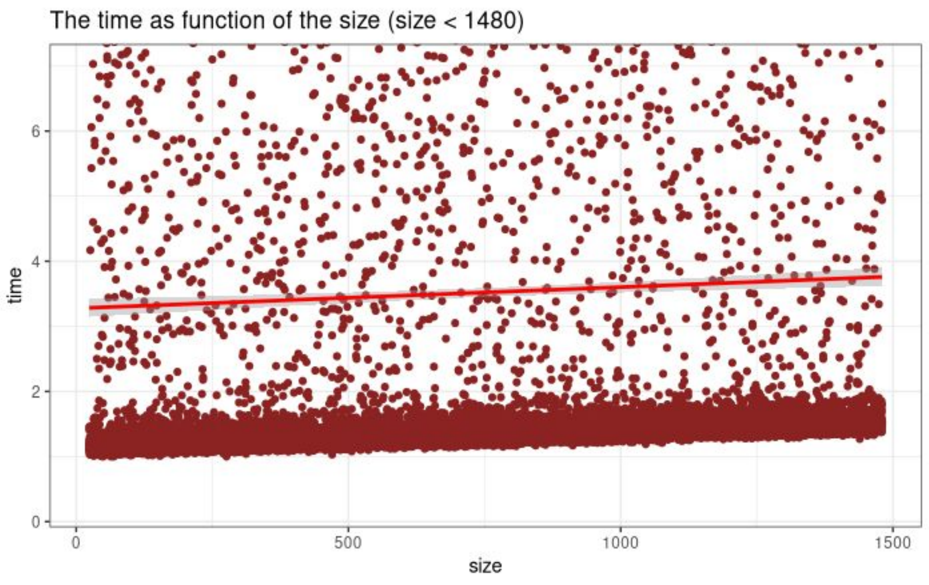
III. Analysing the first sub dataset (size smaller then 1480).



~~~
```{r}
library(ggplot2)
df1 = subset(df,size<=1480&size>=0)
ggplot(data = df1, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw() + coord_cartesian(ylim = c(0.25,7)) +
ggtitle("The time as function of the size (size < 1480)")

```
~~~

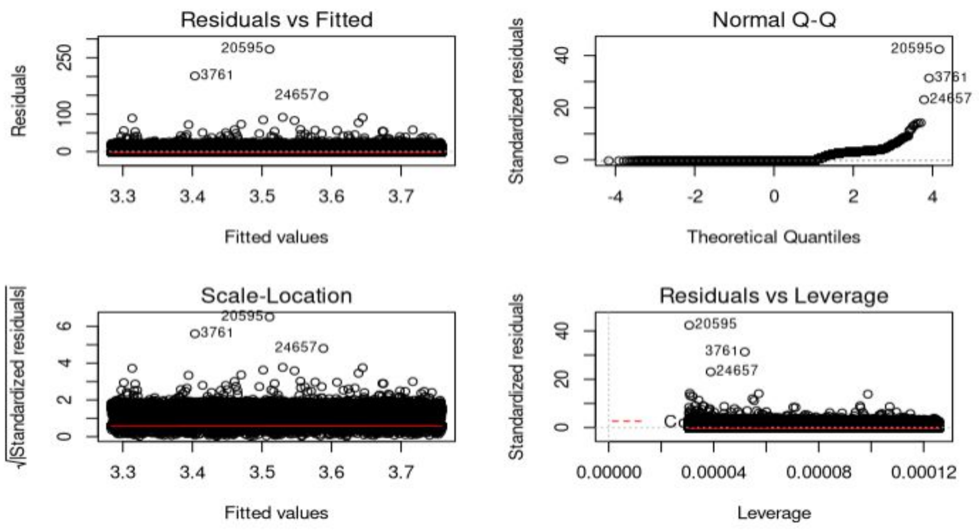


- The figure represent the plot of the linear regression for the dataset with packet size smaller than 1480.
  It's not clear so for this reason we are going to zoom on the interval from 0.25ms to 7 ms.



- Now we can see clearly that most of the paquets have a corresponding time that is close 1.2 ms.

```{r}
mode1 <- lm(data=df1, time~size)
par(mfrow=c(2,2)); plot(mode1); par(mfrow=c(1,1))
```



- **Checking Hypothesis:**

In statistics we start with defining some hypothesis and we work with them then after we finish the calculations we analyse the results and check if the hypothesis are verified or not.
Residuals Vs Fitted : using this plot we check the the homoscedasticity of the model. What we looking at now is the error which should not depend on the time. then if the model is correct we should see something with a quadratic or a heteroscedastic . It's exactly what we see here a uniformly distributed data.
Normality Q_Q plot: if the first hypothesis was not verified the other ones will have no sense. using the Q-Q plot we check if the noise is a normal distribution so we are comparing what we have with a theoretical distribution. In our case we do not have a perfect distributed error but we do not need to have one for linear regression.
Residuals vs Leverage: the leverage tells you what impact a point will have on your predictions and your model. In the other hand it indicates what's wrong with your measurements.

- **Checking the Model :**

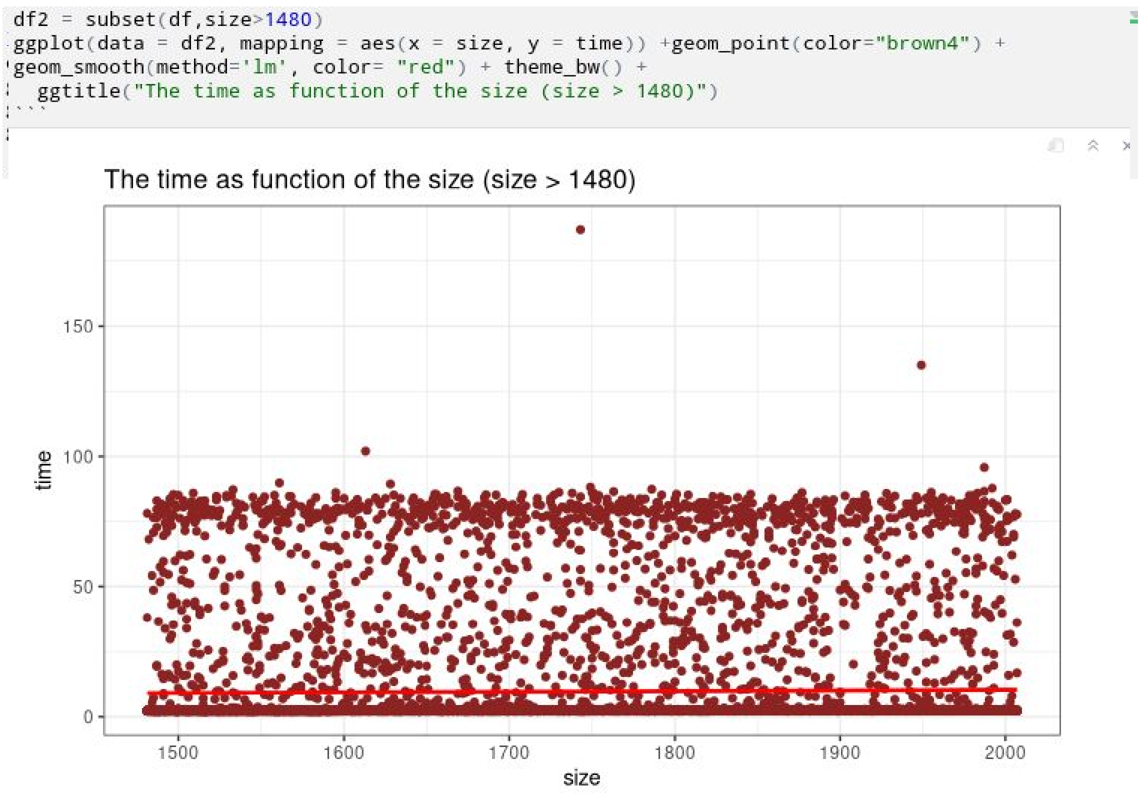
```{r}
summary(mode1)
```

```
Call:
lm(formula = time ~ size, data = df1)

Residuals:
    Min      1Q  Median      3Q     Max 
 -2.475  -2.247  -2.189  -2.088 272.489 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.276e+00  7.231e-02  45.301  < 2e-16 ***
size        3.263e-04  8.497e-05   3.841 0.000123 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.432 on 32665 degrees of freedom
Multiple R-squared:  0.0004514,	Adjusted R-squared:  0.0004208 
F-statistic: 14.75 on 1 and 32665 DF,  p-value: 0.000123
```

The summary give us a lot of information about the model.
Intercept : 3.276 taking in consideration the confidence interval wich we can calculate using the standard error. For 95% Confidence interval : 3.276 ± 2*7.231 10^(-2).
Capacity = 1/intercept (bytes/s)
Slope: 3.273e-04 ± 8.497e-05. Latency= slope(ms)
Degree of freedom: having N sample means we have N observation then N degree of freedom. but in linear regression we have some constraints defined by the model which decrease the degree of freedom. Increasing the number of parameters in the model means that you are reducing the degree of freedom. In this experiment we have a big number of samples and less parameters for this reason the degree of freedom is Big.
R squared: a low R squared means that we have an important noise or a bad model. in this case we have a small R squared which means neither our model is not good or we have a lot of noise.
Pr(>|t|): is computing the probability of a value to not be absurd. So we compute the value we want and we see if it is representative in the distribution we have (t). we want this value to be as low as possible.

Now we will do the same thing for the second interval.

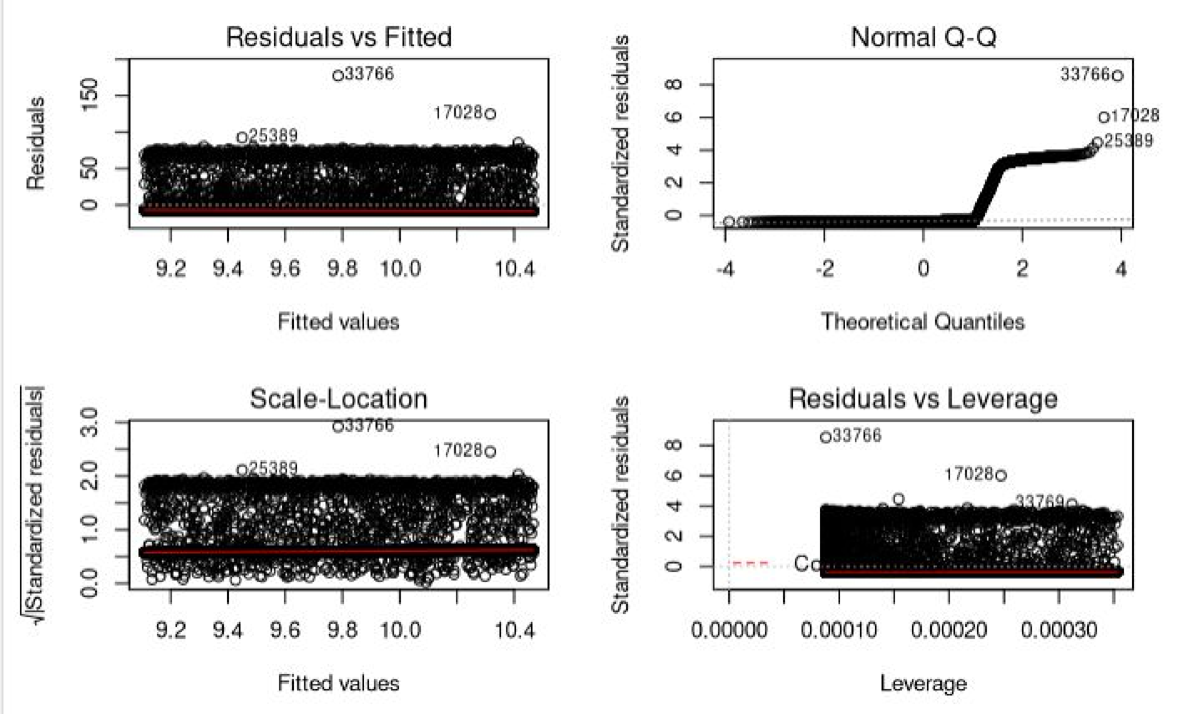
IV. Analysing the second sub dataset (size greater then 1480).



```
df2 = subset(df,size>1480)
ggplot(data = df2, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw() +
  ggtitle("The time as function of the size (size > 1480)")
```
```

- The figure represent the plot of the linear regression for the dataset with packet size greater then 1480.
```

Checking the Hypothesis

~~~
```{r}
mode2 <- lm(data=df2, time~size)
par(mfrow=c(2,2)); plot(mode2); par(mfrow=c(1,1))
```
~~~



Similar to what have been said for the first interval.

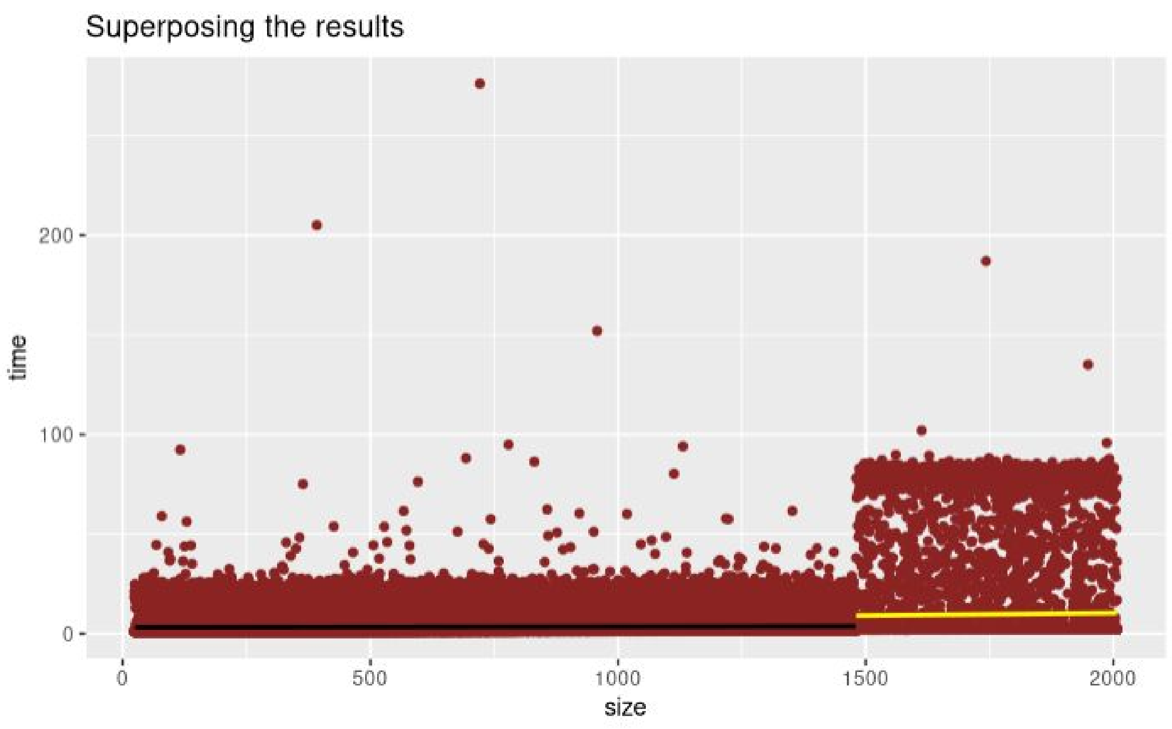
Checking the model

~~~
```{r}
summary(mode2)
```
~~~

```
Call:
lm(formula = time ~ size, data = df2)

Residuals:
    Min      1Q  Median      3Q     Max
 -8.276  -7.729  -7.354  -6.999 177.215

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.289833   2.244269   2.357   0.0184 *
size        0.002579   0.001281   2.012   0.0442 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.74 on 11367 degrees of freedom
Multiple R-squared:  0.0003562, Adjusted R-squared:  0.0002682
F-statistic:  4.05 on 1 and 11367 DF,  p-value: 0.0442
```

Intercept : For 95% Confidence interval : 5.289833 ± 2*2.244269.
Slope: 0.002579 ± 0.001281.
Degree of freedom: The same thing we have a big freedm degree because we have a lot o f samples and the number of the parameters in the model still the same.
R squared: we have always a small R squared so we still have the same problem either our model is bad or we have a lot of noise.
Pr(>|t|): we have a bad probability of the intercept and the slope to be absurde.

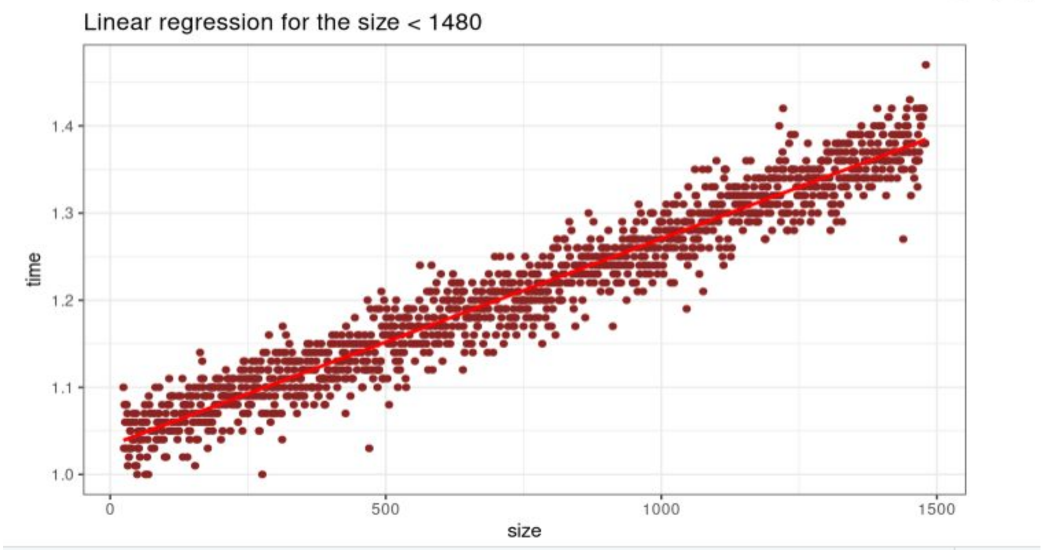
V. subplot of both datasets linear regression in one figure

~~~r
```{r}
ggplot(data = df, mapping = aes(x =size, y = time)) + geom_point(col="brown4")+
geom_smooth(method='lm', data=df1, color= "black")+ geom_smooth(method='lm', data=df2, color=
"yellow") + ggtitle("Superposing the results")
~~~
```


```

## VI. Filtering the Data taking the smallest times per size

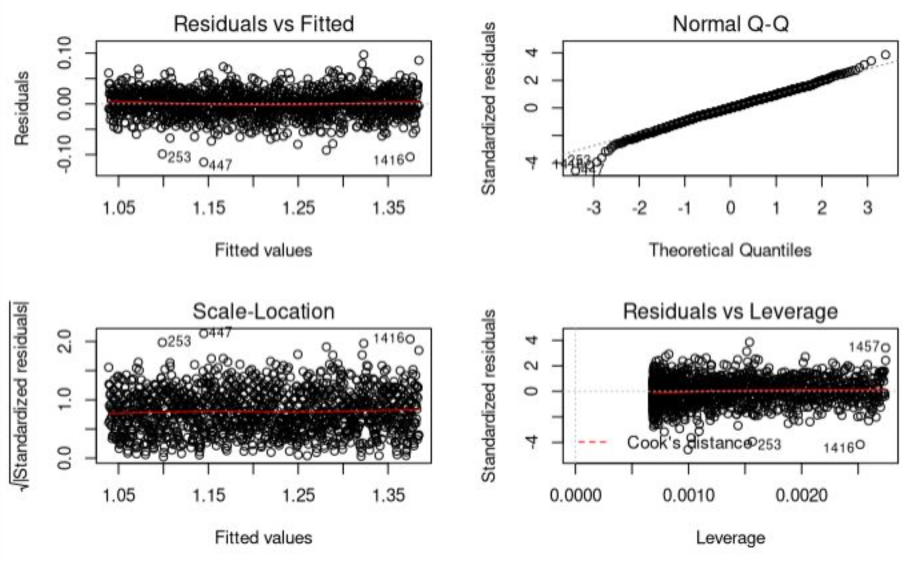
The first data set has a strong variability and it is asymmetric so the regression of the mean transmission time can be considered as irrelevant. Therefore we are going to filter the dataset taking only the minimum time per size. like that the number of samples and the variability of the noise is gonna be reduced.

```
df1_min = unique(aggregate(time~size,df1,min))
ggplot(data = df1_min, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw()+
  ggtitle("Linear regression for the size < 1480")
```



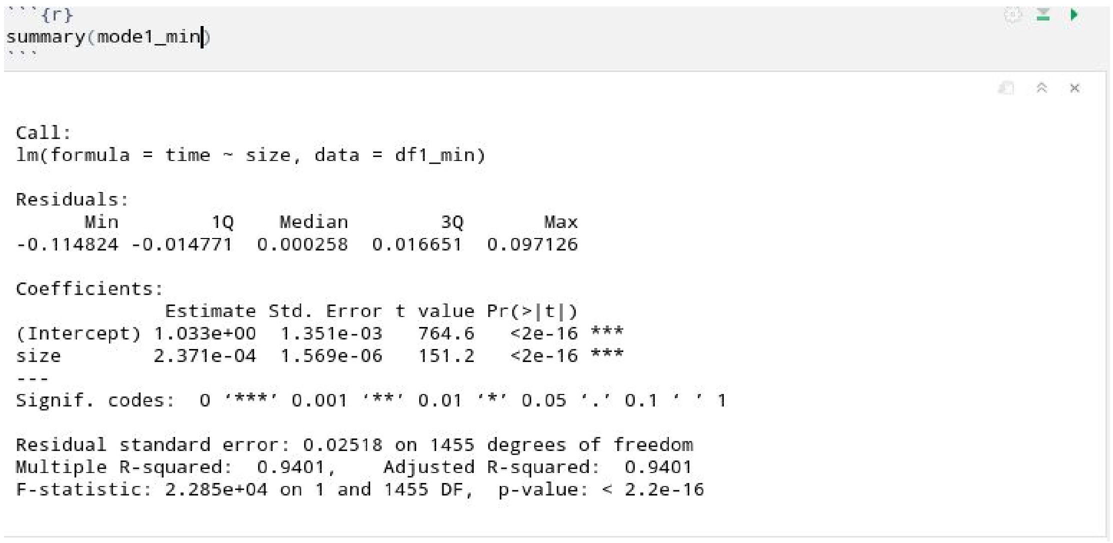
- The plot now shows that the samples are perfectly linear and also symmetric.
- **Checking the hypothesis**

```{r}
mode1_min <- lm(data=df1_min, time~size)
par(mfrow=c(2,2)); plot(mode1_min); par(mfrow=c(1,1))
```



Residuals Vs Fitted : we have a symmetric homoscedastic model.
Normality Q_Q plot: the noise has a normal distribution.

- **Checking the model.**

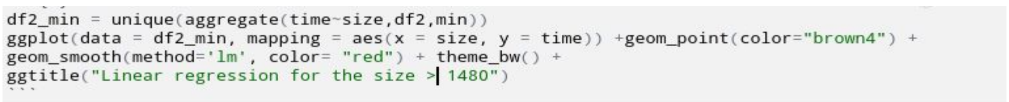


~~~
```{r}
summary(mode1_min)
```
~~~

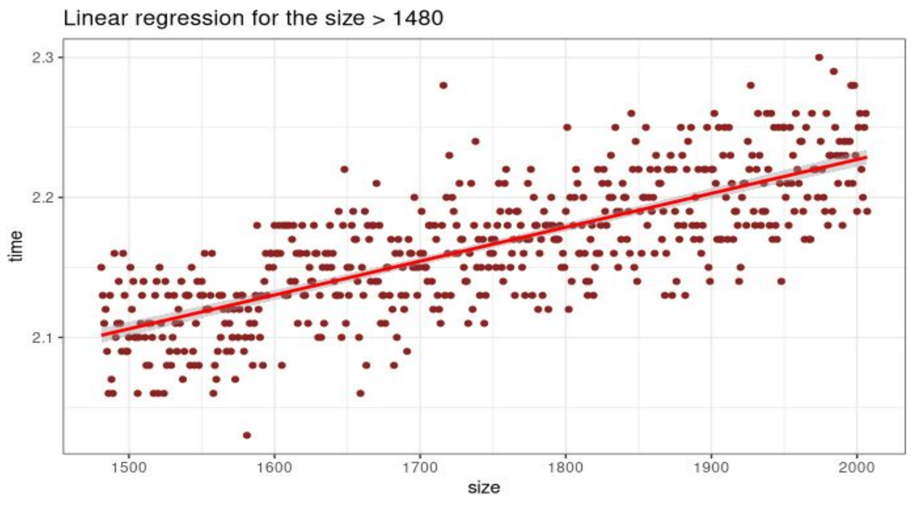
```
Call:
lm(formula = time ~ size, data = df1_min)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.114824 -0.014771  0.000258  0.016651  0.097126 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.033e+00  1.351e-03   764.6   <2e-16 ***
size        2.371e-04  1.569e-06   151.2   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.02518 on 1455 degrees of freedom
Multiple R-squared:  0.9401,	Adjusted R-squared:  0.9401 
F-statistic: 2.285e+04 on 1 and 1455 DF,  p-value: < 2.2e-16
```

Intercept : For 95% Confidence interval : 1.033 ± 1.341e-03.
Slope: 2.374e-04 ± 1.5691.341e-06.
Degree of freedom: The degree of freedom is smaller because we have less measurements.
R squared: R squared is high so we have less noise and the model is good too.
Pr(>|t|): we have a good probability of the intercept and the slope to be absurde. also the p-value is small so we can trust this values.

Applying the same thing for the second interval.

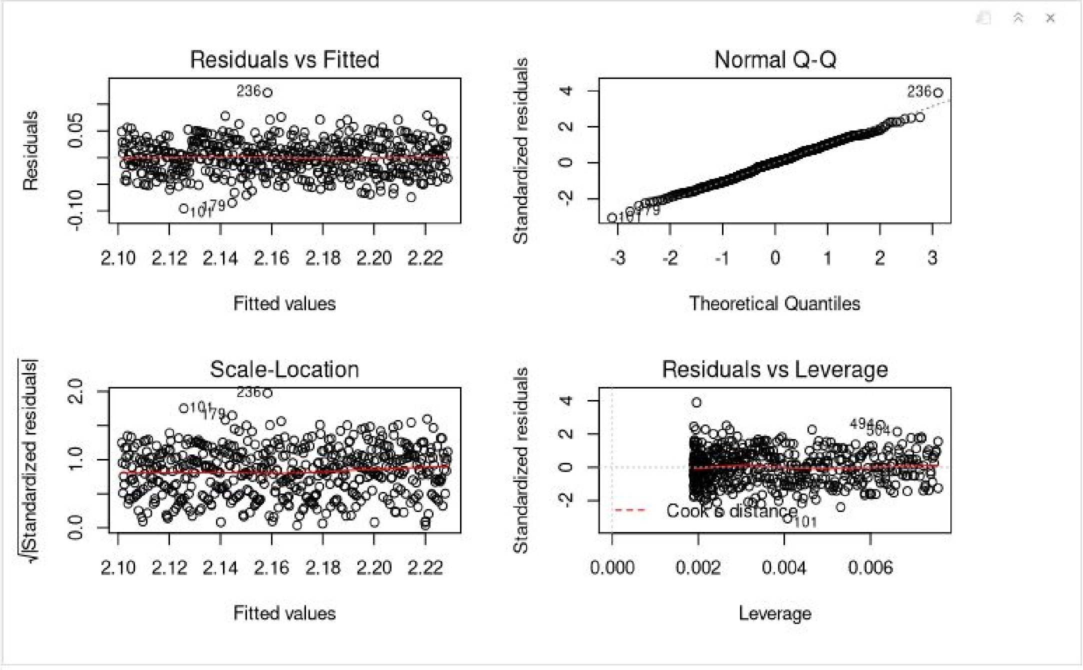


```
df2_min = unique(aggregate(time~size,df2,min))
ggplot(data = df2_min, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw() +
ggtitle("Linear regression for the size > 1480")
```



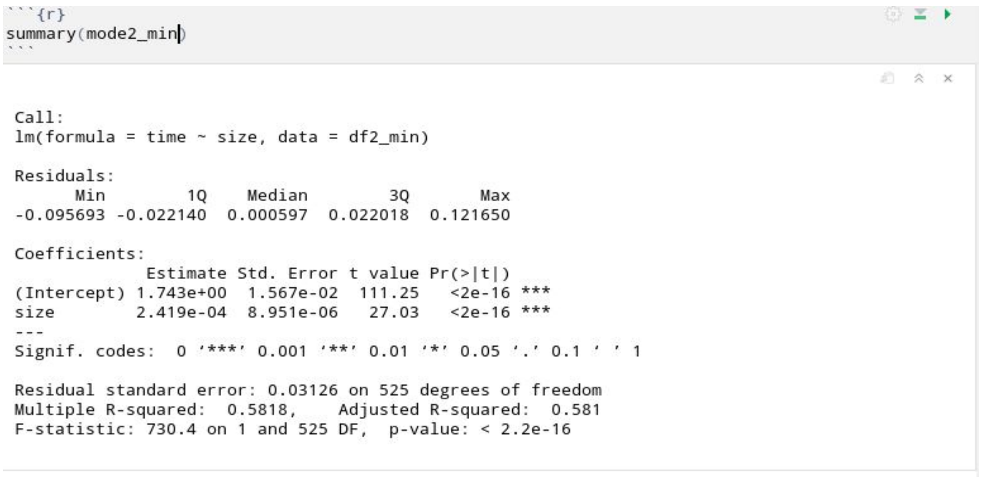
- **Checking the hypothesis.**

```
mode2_min <- lm(data=df2_min, time~size)
par(mfrow=c(2,2)); plot(mode2_min); par(mfrow=c(1,1))
```



we have similar results to what we had on the first interval.

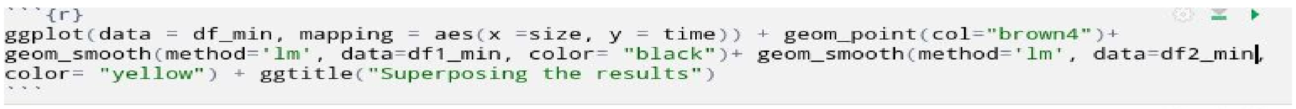
- **Checking the model**



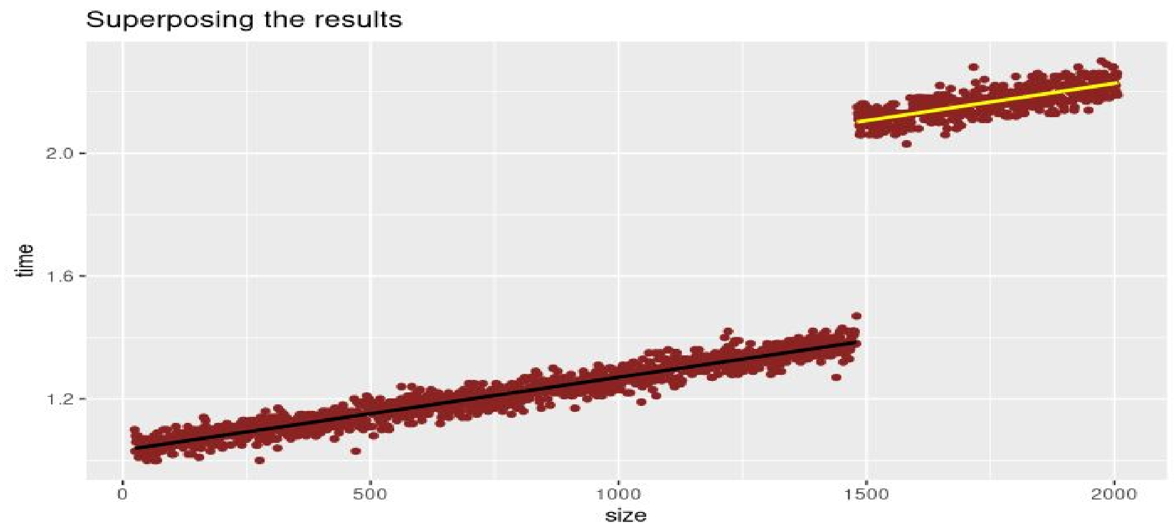
~~~
```{r}
summary(mode2_min)
```

Call:
lm(formula = time ~ size, data = df2_min)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.095693 -0.022140  0.000597  0.022018  0.121650 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.743e+00  1.567e-02  111.25   <2e-16 ***
size        2.419e-04  8.951e-06   27.03   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03126 on 525 degrees of freedom
Multiple R-squared:  0.5818,	Adjusted R-squared:  0.581 
F-statistic: 730.4 on 1 and 525 DF,  p-value: < 2.2e-16
~~~

Intercept : For 95% Confidence interval : 1.743e00 ± 1.576e-02.
Slope: 2.419e-04 ± 8.951e-06.
Degree of freedom: The degree of freedom is smaller because we have less measurements.
R squared: R squared is small because our model is bad it does not take in consideration the fragmentation of the packets and the overhead that it will be created.
Pr(>|t|): we have a good probability of the intercept and the slope to be absurde. also the p-value is small so we can trust this values.

- **Superposing the both intervals**



~~~
```{r}
ggplot(data = df_min, mapping = aes(x =size, y = time)) + geom_point(col="brown4")+
geom_smooth(method='lm', data=df1_min, color= "black")+ geom_smooth(method='lm', data=df2_min,
color= "yellow") + ggtitle("Superposing the results")
```
~~~

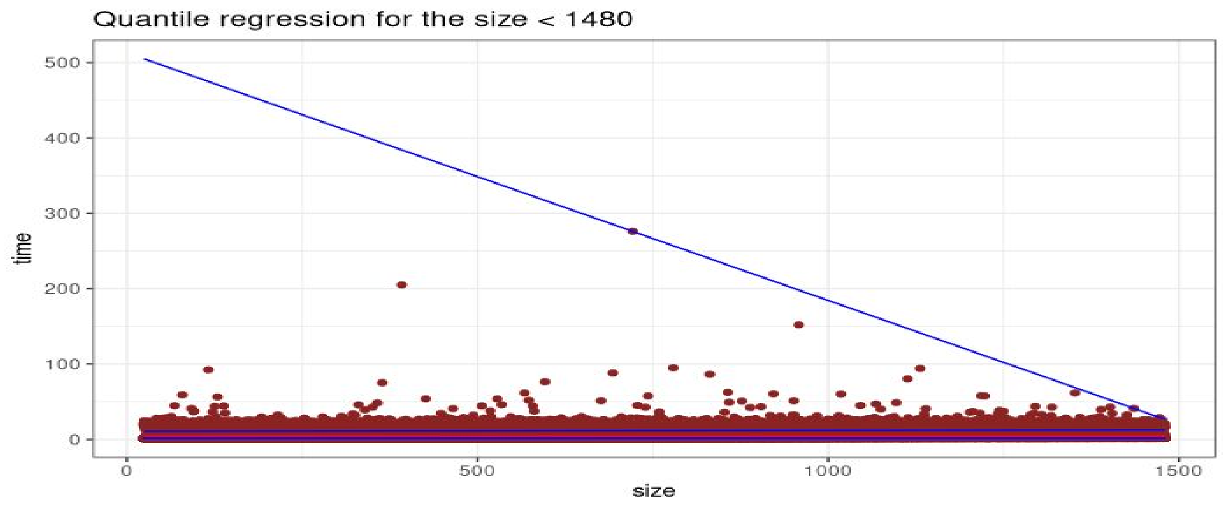


Conclusion: filtering and taking the minimum time by size seems to be a good solution. but in fact it is not because: Firstable we are ignoring some measurements which they can be really important for our model. secondly those measurements we chose (the minimum time) maybe they are wrong because it is a physical system and we can not have a perfect experiment so we may have some external impact from the environment on some values.

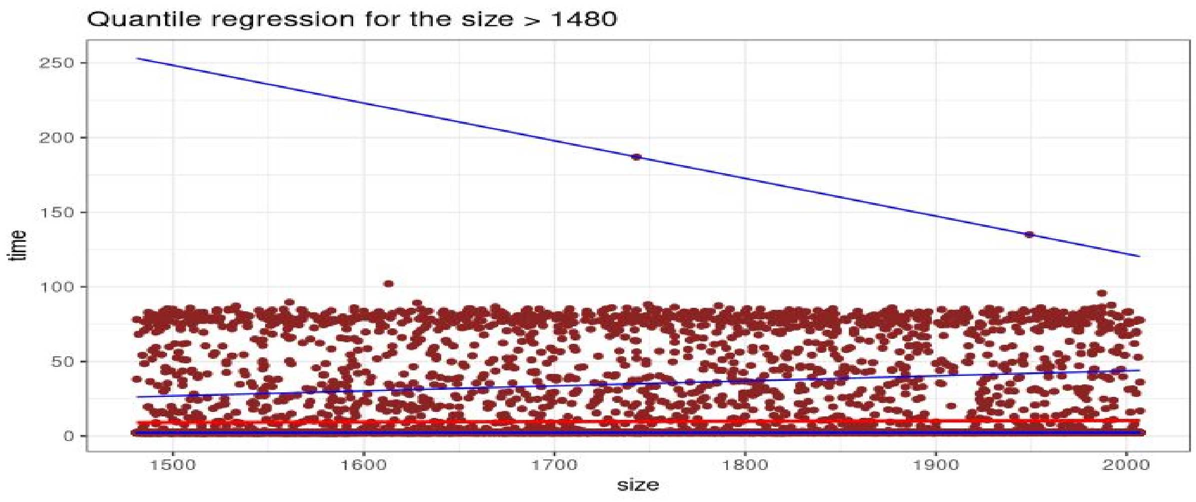
VII. Quantile regression

One of the limits of the linear regression is that the parameters that are representing the model are constant and calculated by the mean. It is not suited for non linear models or having some extreme samples. In the other hand quantile regression is good for this type of data and that's why we are using it here. we are using 10 quantiles.

~~~
```{r}
q10 <- seq(0, 1, by = 0.1)
ggplot(data = df1, mapping = aes(x = size, y = time)) +
  geom_point(color="brown4") +
  geom_smooth(method='lm', color= "red") + theme_bw() +
  geom_quantile (quantiles = q10, color="blue")+
  ggtitle("Quantile regression for the size < 1480")
```
~~~



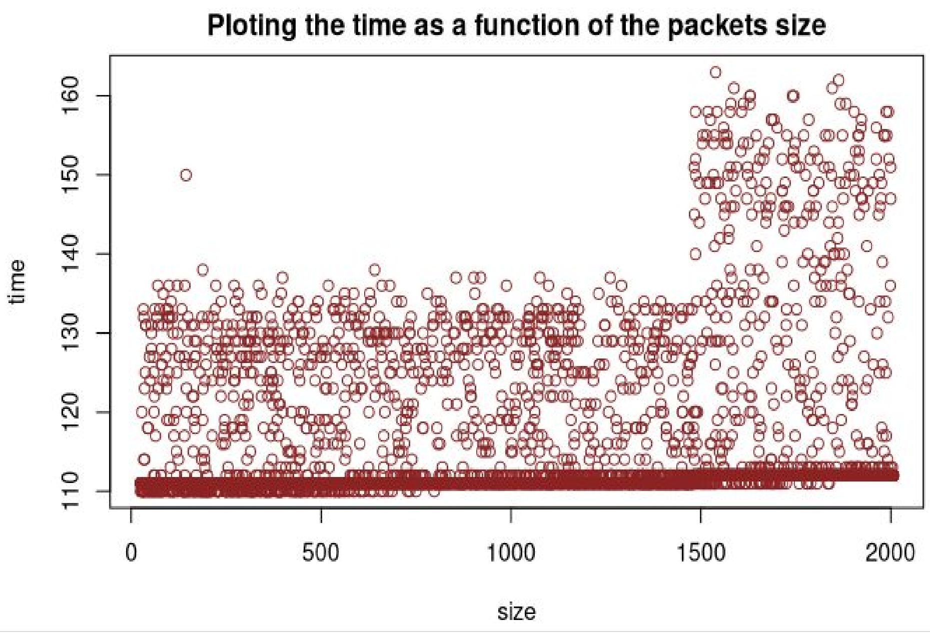
~~~
```{r}
q10 <- seq(0, 1, by = 0.1)
ggplot(data = df2, mapping = aes(x = size, y = time)) +
  geom_point(color="brown4") +
  geom_smooth(method='lm', color= "red") + theme_bw() +
  geom_quantile (quantiles = q10, color="blue") +
  ggtitle("Quantile regression for the size > 1480")

```
~~~



we have differents lines taking inconsideration the impact of the size for different levels of the time from the smallest to the bigger one. the quantile regression can show us what's wrong with our experiment and the measurements that we need to ignore.

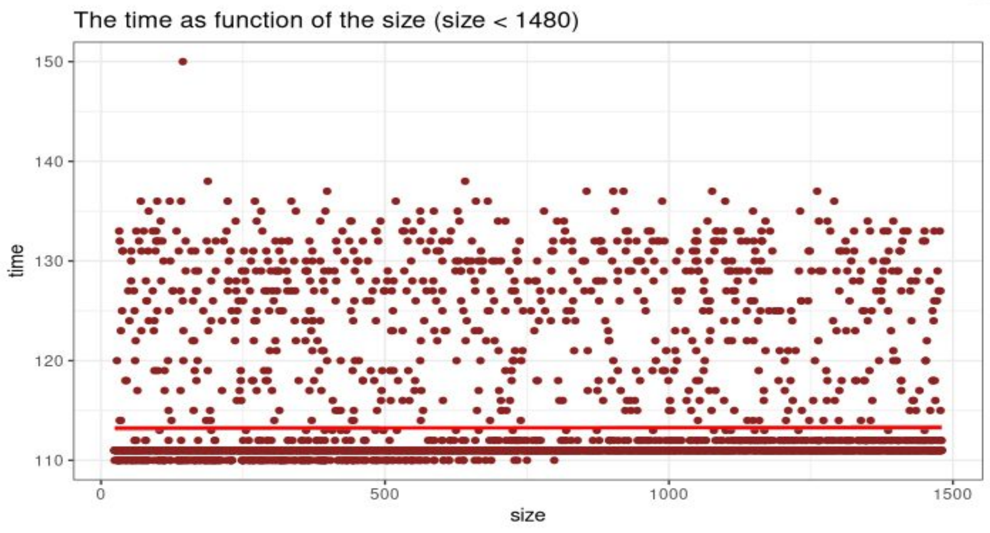
VIII. Second Dataset

- **Prepare the Data**

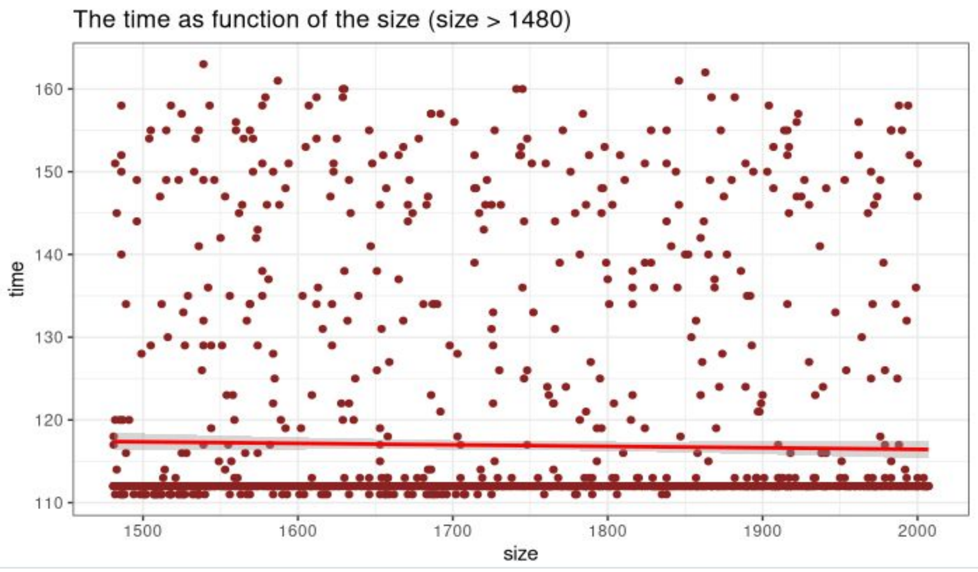
~~~
```{r}
df_1 = read.table('stackoverflow.log', sep=' ' , na.strings ="" , header=F , fill = TRUE )
~~~
```

```
Clean the data:
```{r}
df_1 = df_1 %>% select(V1, V2, V9)
library(tidyr)
line_NA  = apply(df_1 , 1 , function(x) any(is.na(x)))
df_1 = df_1 %>% drop_na()
colnames(df_1)=c('date' ,'size' , 'time' )
convertdate = function(date)
  gsub("[^0-9.]", "", date)


df_1$date = as.numeric(sapply(df_1$date , convertdate))

convertTime = function(time)
  gsub("[^0-9.]", "", time)

df_1$time = as.numeric(sapply(df_1$time , convertTime))
```
```

- **Plot the data**



we still have two patterns so we are going to divide the data to two intervals.
```

- **Divide the data**

~~~
```{r}
library(ggplot2)
df_1.1 = subset(df_1,size<=1480&size>=0)
ggplot(data = df_1.1, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw()  + ggtitle("The time as function of the
size (size < 1480)")
```
~~~



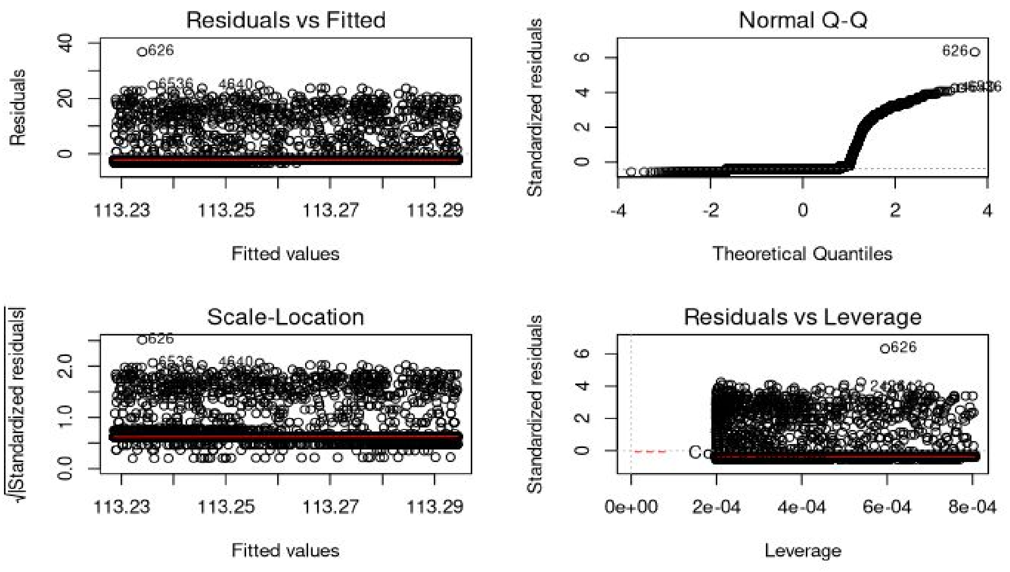
~~~
```{r}
df_1.2 = subset(df_1,size>1480)
ggplot(data = df_1.2, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw() +
  ggtitle("The time as function of the size (size > 1480)")
```
~~~



now we have less measurements but the line is really bad.

- **Checking hypothesis**

```{r}
mode_1.1 <- lm(data=df_1.1, time~size)
par(mfrow=c(2,2)); plot(mode_1.1); par(mfrow=c(1,1))
```



we have more variability and the data is more asymmetric then the first data set.

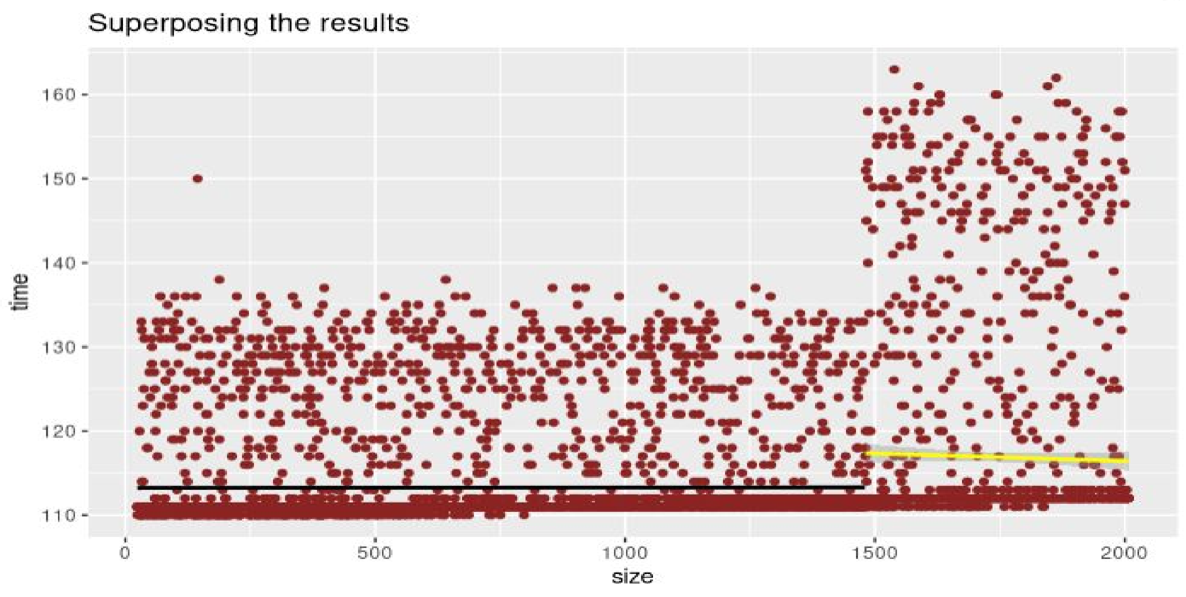
- **Checking the model**

```
Call:
lm(formula = time ~ size, data = df_1.1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.264 -2.276 -2.255 -2.232 36.766 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.132e+02  1.655e-01 684.253   <2e-16 ***
size        4.521e-05  1.938e-04   0.233    0.816    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.818 on 5013 degrees of freedom
Multiple R-squared:  1.085e-05,	Adjusted R-squared:  -0.0001886 
F-statistic: 0.05442 on 1 and 5013 DF,  p-value: 0.8156


Call:
lm(formula = time ~ size, data = df_1.2)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.382 -5.113 -4.789 -4.468 45.721 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 120.053588   3.188867  37.648   <2e-16 ***
size         -0.001803   0.001827  -0.987    0.324    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.88 on 1807 degrees of freedom
Multiple R-squared:  0.0005387,	Adjusted R-squared:  -1.444e-05 
F-statistic: 0.9739 on 1 and 1807 DF,  p-value: 0.3238
```

we have bad regression model with a lot of noise .

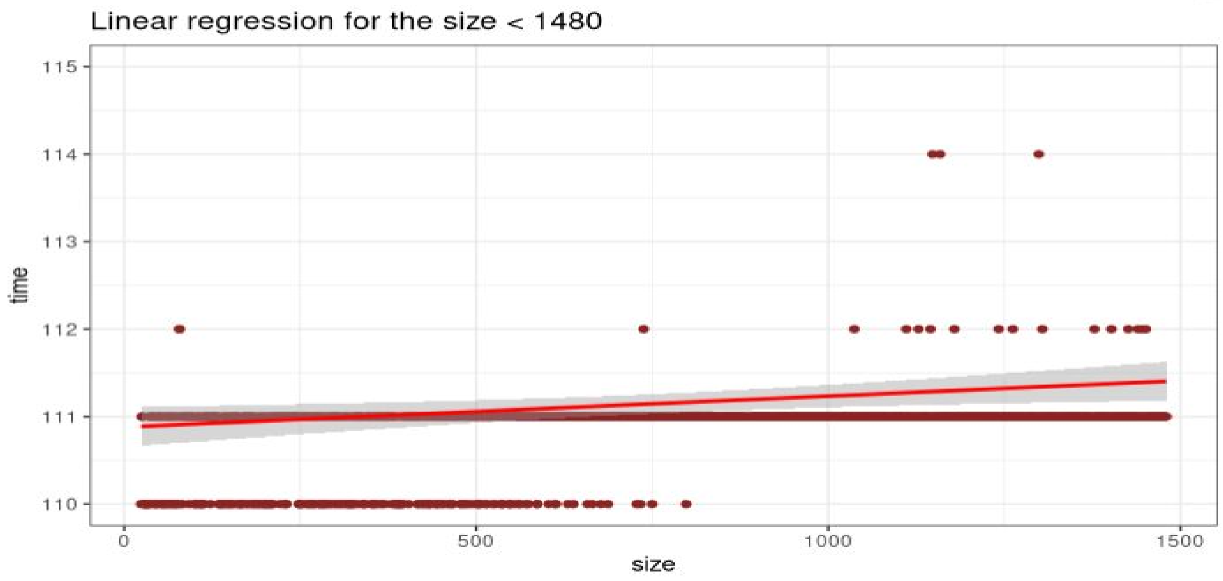
- small R squared.
- Big p-value**.**

```{r}
ggplot(data = df_1, mapping = aes(x =size, y = time)) + geom_point(col="brown4")+
geom_smooth(method='lm', data=df_1.1, color= "black")+ geom_smooth(method='lm', data=df_1.2,
color= "yellow") + ggtitle("Superposing the results")
```

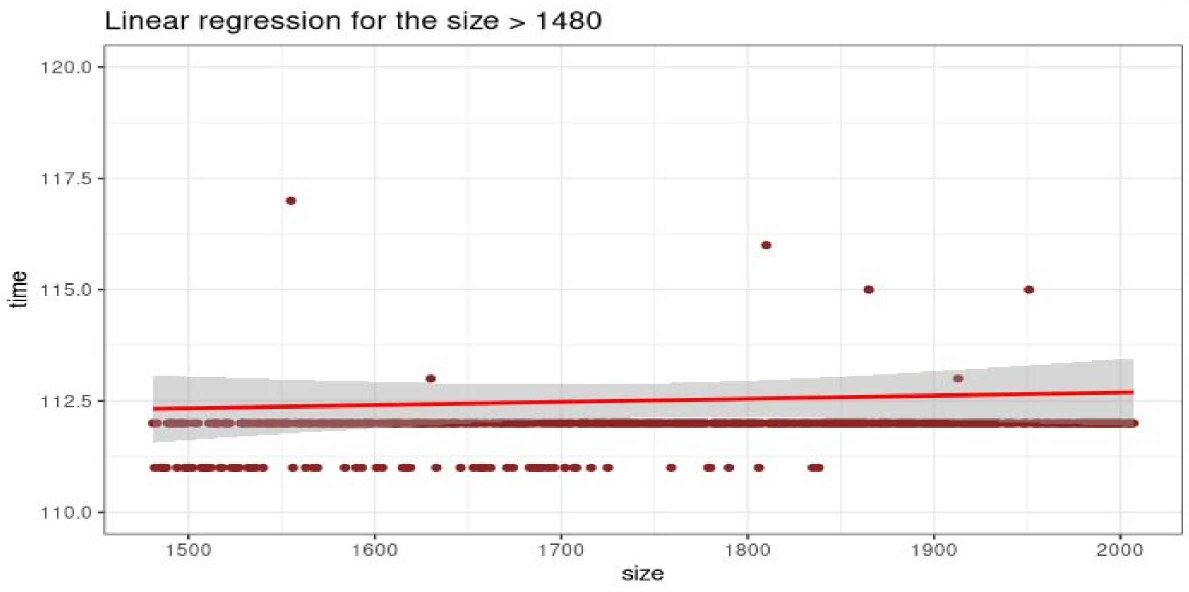


- **Taking the minimum of the time per size.**

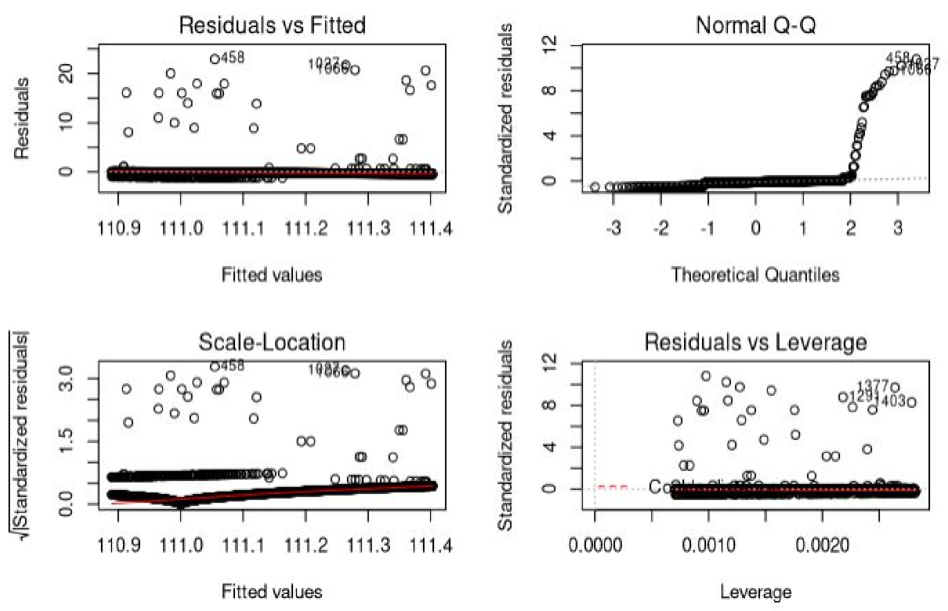
```
df_1.1_min = unique(aggregate(time~size,df_1.1,min))
ggplot(data = df_1.1_min, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw()+ coord_cartesian(ylim = c(110,115))+
  ggtitle("Linear regression for the size < 1480")
```



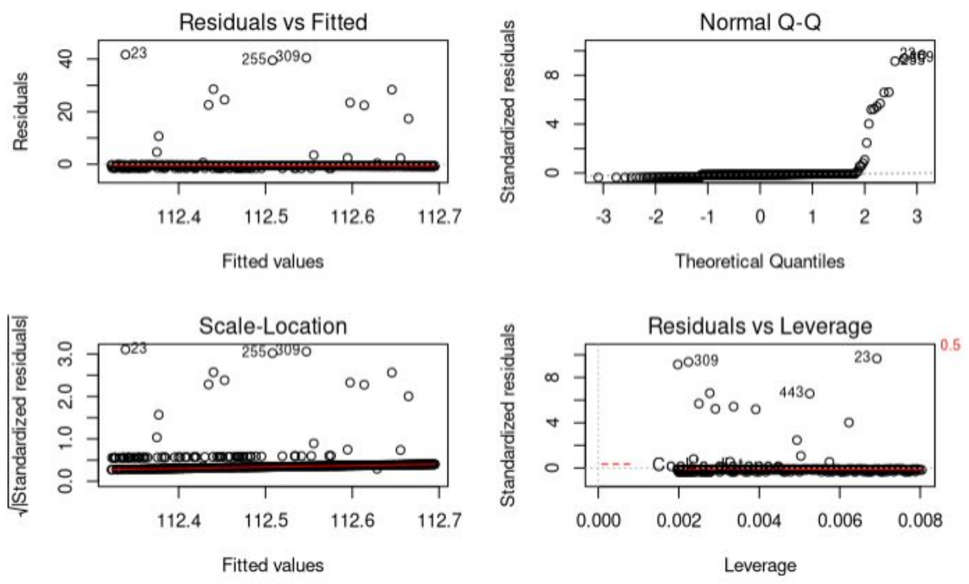
```
df_1.2_min = unique(aggregate(time~size,df_1.2,min))
ggplot(data = df_1.2_min, mapping = aes(x = size, y = time)) +geom_point(color="brown4") +
geom_smooth(method='lm', color= "red") + theme_bw() + coord_cartesian(ylim = c(110,120)) +
ggtitle("Linear regression for the size > 1480")
```

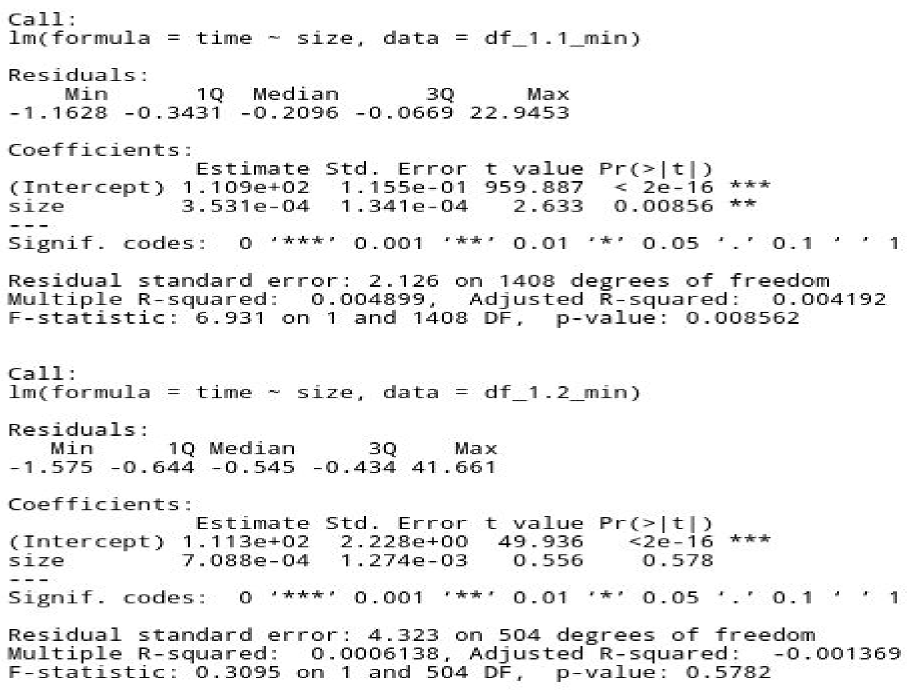


```{r}
mode_1.1_min <- lm(data=df_1.1_min, time~size)
par(mfrow=c(2,2)); plot(mode_1.1_min); par(mfrow=c(1,1))
```



```{r}
mode_1.2_min <- lm(data=df_1.2_min, time~size)
par(mfrow=c(2,2)); plot(mode_1.2_min); par(mfrow=c(1,1))
```





```
Call:
lm(formula = time ~ size, data = df_1.1_min)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.1628 -0.3431 -0.2096 -0.0669 22.9453 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.109e+02  1.155e-01 959.887  < 2e-16 ***
size        3.531e-04  1.341e-04   2.633  0.00856 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.126 on 1408 degrees of freedom
Multiple R-squared:  0.004899,	Adjusted R-squared:  0.004192 
F-statistic: 6.931 on 1 and 1408 DF,  p-value: 0.008562


Call:
lm(formula = time ~ size, data = df_1.2_min)

Residuals:
   Min     1Q Median     3Q    Max 
-1.575 -0.644 -0.545 -0.434 41.661 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.113e+02  2.228e+00  49.936   <2e-16 ***
size        7.088e-04  1.274e-03   0.556    0.578    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.323 on 504 degrees of freedom
Multiple R-squared:  0.0006138,	Adjusted R-squared:  -0.001369 
F-statistic: 0.3095 on 1 and 504 DF,  p-value: 0.5782
```

Applying the linear regression to the second dataset gives bad results because the model we are applying take only the size in consideration. but her in this case we have different environment because we have more load. which means that there is another parameters to take in consideration to build the model.

Conclusion and feedback

Doing this exercise was really helpful firstabale to apply and practice the theoretical knowledge we had in the lectures.
As a conclusion i think the first thing is that the model was very simple and it didn’t take a lot of parameters in consideration as it was said in the exercice subject description. To have a better R squared we can add more parameters but we need to compromise between overfitting the model and simplicity.
in addition to this here we can see the importance of the design of experiment especially choosing the way of doing measurements. i think personally that here we had a lot of data which make the freedom degree so big and also it gave us a lot of noise. in the other hand if we do less measurements we will have a weak model that can miss some important information.
So as we see here or as i see now doing statistics is dependable because you have to compromise between a lot of things. therefore you need to adapt your approach to the problem you have in hand.