Latency and capacity estimation for a network connection from asymmetric measurements

Libraries used

```
suppressMessages(library(stats  , warn.conflicts = FALSE))
suppressMessages(library(SparseM  , warn.conflicts = FALSE))
suppressMessages(library(quantreg , warn.conflicts = FALSE ))
suppressMessages(library(lubridate , warn.conflicts = FALSE))
suppressMessages(library(dplyr , warn.conflicts = FALSE))
suppressMessages(library(ggplot2  , warn.conflicts = FALSE))
suppressMessages(library(tidyr , warn.conflicts = FALSE))
```

Reading Data, if na.string="" we will replace the field by NA and we will take 3 columns V1, V2, V9 columns coresponding to time, size, and time transmission

```
df = read.table('liglab2.log', sep=' ' , na.strings = ""  , header=F , fill = TRUE  )
df = df %>% select(V1, V2, V9)
df[10:19 , ]
```

```
##                      V1   V2        V9
## 10 [1421761684.122687] 1422 time=19.5
## 11 [1421761684.344135] 1180 time=18.0
## 12 [1421761684.566271]  999 time=18.8
## 13 [1421761684.770828]   21      <NA>
## 14 [1421761684.998504] 1020 time=24.3
## 15 [1421761685.205172]   71 time=3.45
## 16 [1421761685.414106]   34 time=5.85
## 17 [1421761685.620117] 1843 time=2.31
## 18 [1421761685.824949]  407 time=1.14
## 19 [1421761686.029177]  356 time=1.10
```

As we can see we have cells in some rows with NA value

Cleaning the dataframe by droping all rows with at least one cell with NA

```
df = df %>% drop_na()
line_NA  = apply(df , 1 , function(x) any(is.na(x)))
df[line_NA ,  ]
```

```
## [1] V1 V2 V9
## <0 rows> (or 0-length row.names)
```

As we can see above,after removing the rows that contains at least 1 cell with NA value, we tried to print and we had 0 rows contains NA, now we have our data cleaned.

Changing column names for better View

```
colnames(df)=c('date' ,'size' , 'time' )
```

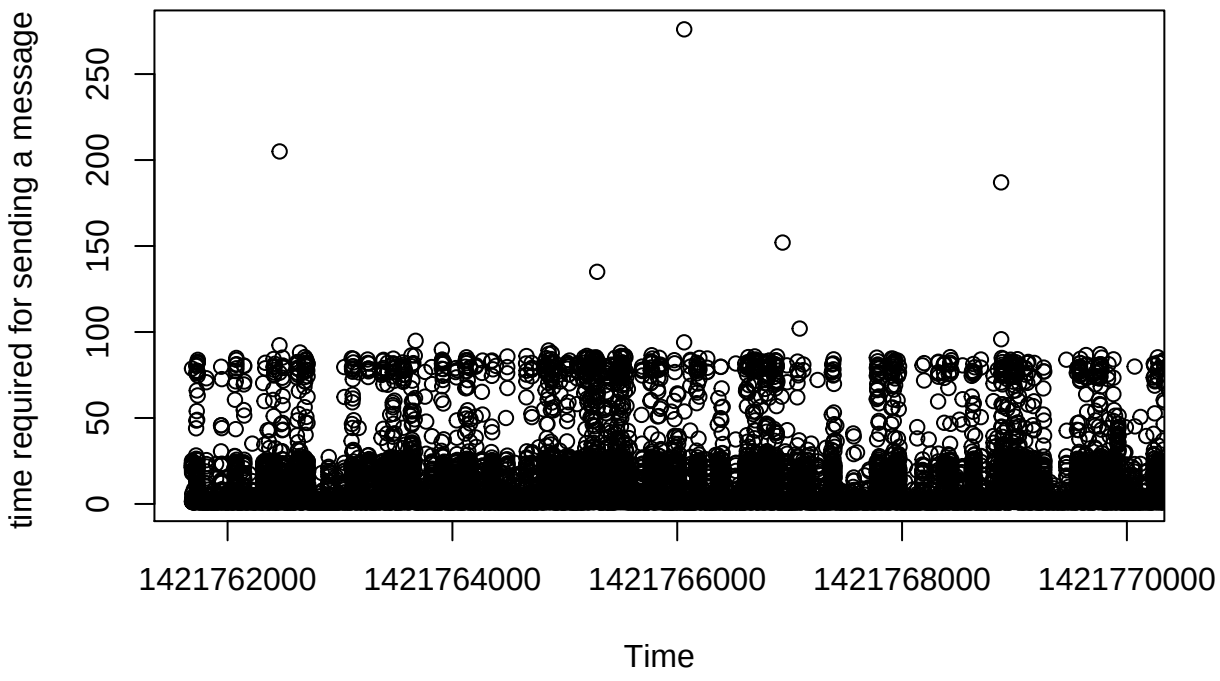
Converting time and transmission time columns datatype to double

```
convertTime = function(time)
  gsub("[^0-9.]", "", time)
df$time = as.numeric(sapply(df$time , convertTime))
options(digits=16)
df$date = as.double(sapply(df$date , convertTime))
head(df)
```

```
##                date size  time
## 1 1421761682.052172  665 22.50
## 2 1421761682.277315 1373 21.20
## 3 1421761682.502054  262 21.20
## 4 1421761682.729257 1107 23.30
## 5 1421761682.934648 1128  1.41
## 6 1421761683.160397  489 21.90
```

option is used to allow the user to set and examine a variety of global "options" which affect the way in which R computes and displays its results. now we have types of time and data is double. Now our data is cleaned and ready to be contucted to analysis.

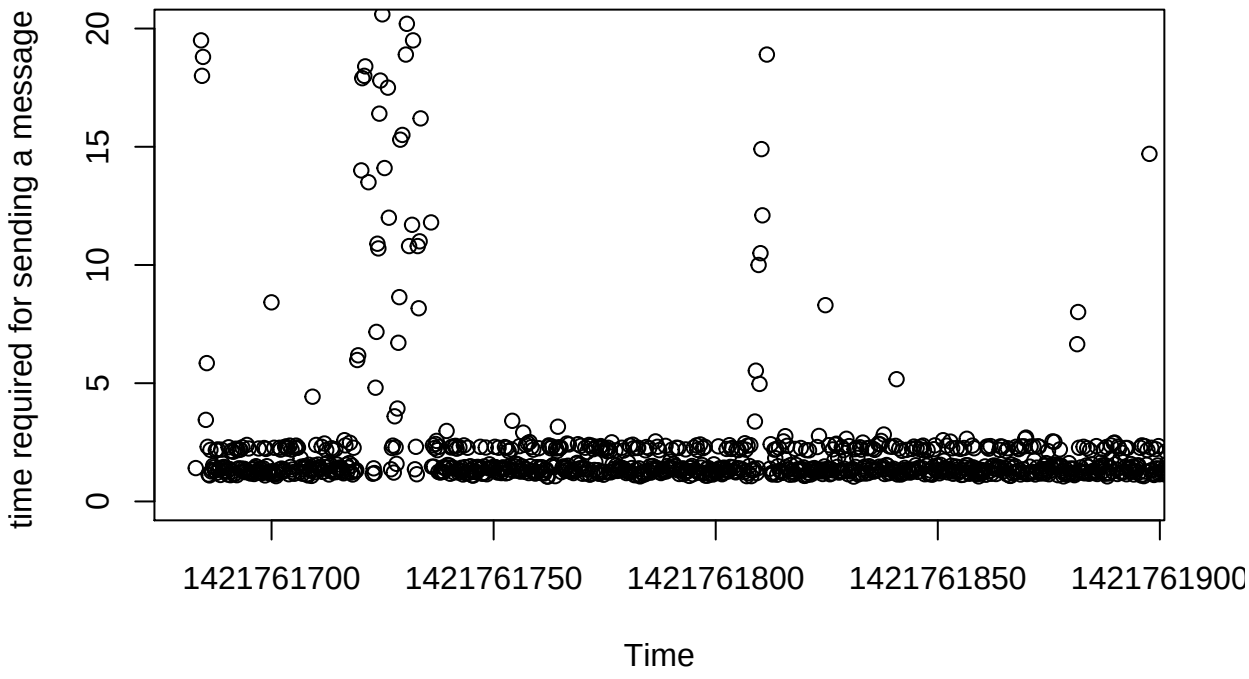
```
plot(df$date , df$time , xlab="Time" ,ylab="time required for sending a message " ,
     xlim=c(1421761682.05217,1421770000.59017) )
```



this plot represent the time transmission on a period

taking a slot to better observation

```
plot(df$date , df$time , xlab="Time" , ylab="time required for sending a message " ,
     xlim=c(1421761682.05217,1421761892.59017) , ylim=c( 0,20 ) )
```

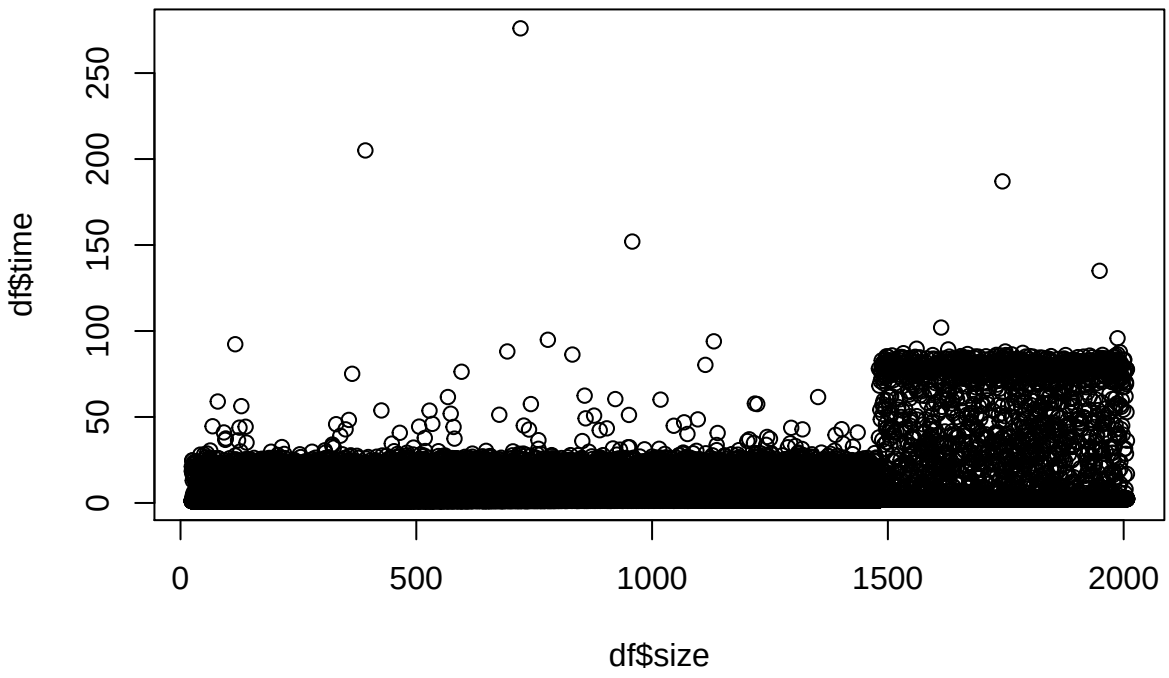


We can here observe that there is a sudden increase in the time required for sending a message . This shows that not only message size affect the the transsmission time of sending the ping packets. (I guess something related to network trafiic affecting the time required for sending a message).

```
#checking the message size in the slot of sudden increase
#plot(df$date , df$size , xlab="Time" , ylab="msg size" , xlim=c(1421761720.05217,1421761725.59017) )
```

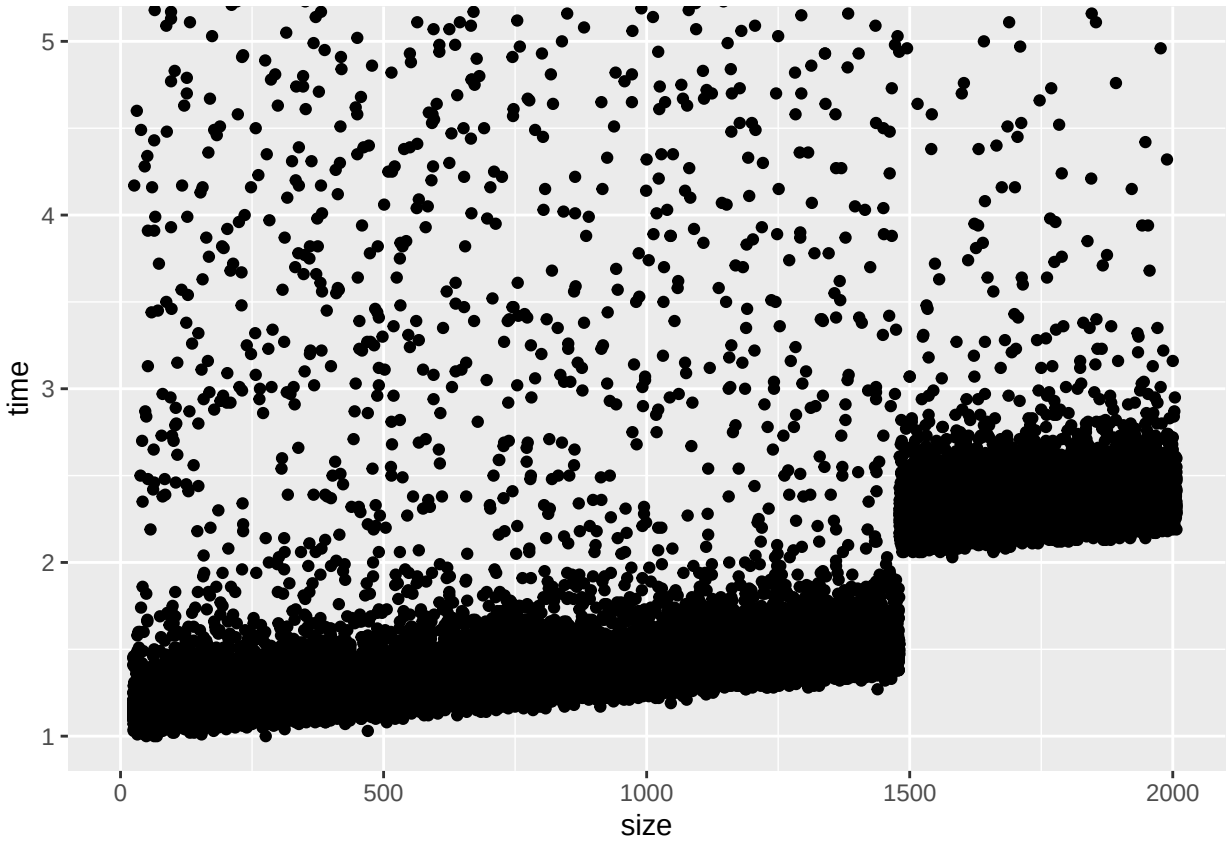
Curve representing the time transmission in terms of message size

```
plot(df$size , df$time)
```



As we can see in the figure once the packet size is greater that ~1500 the behaivoir of time transmissionis changed it start to take more time. here we have 2 cases behaves differently In a standard Ethernet network the MTU is 1500 bytes. The maximum packet size within the frame is 1472 bytes. If packet fragmentation is allowed, a packet can be up to 65,535 bytes. It would just be split into multiple frames and sent. If fragmentation isn't allowed, packet size has to be controlled so that it doesn't get too big to fit within the frame. This is why we have larger time required for sending a message when packet size is more that 1500(not only this factor but size has an effect on the transmssion time as we can see in the figure).

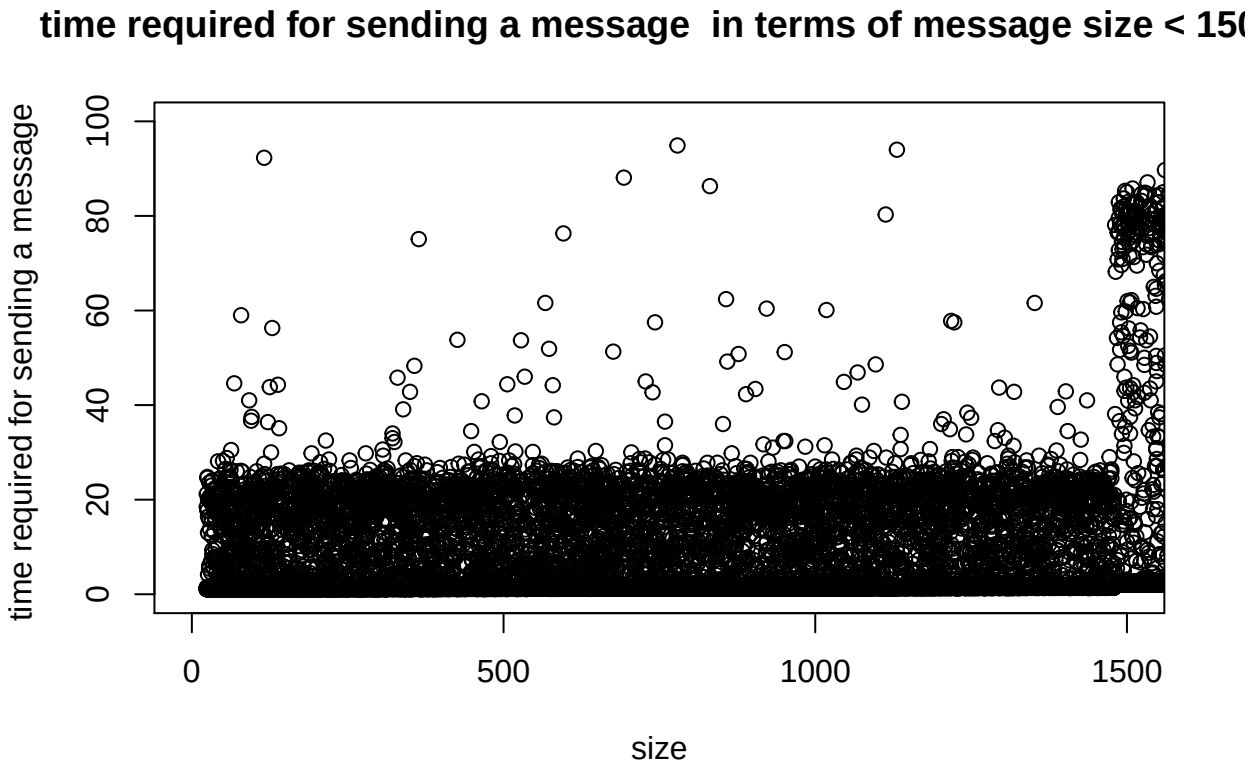
```
ggplot(data = df, mapping = aes(x = size, y = time)) +
    geom_point() +
    coord_cartesian(ylim=c(1, 5) , xlim=c(0 , 2000))
```



we can observe above that the message size actually have an effect on the transmission time, as message size increase transmission time increase.

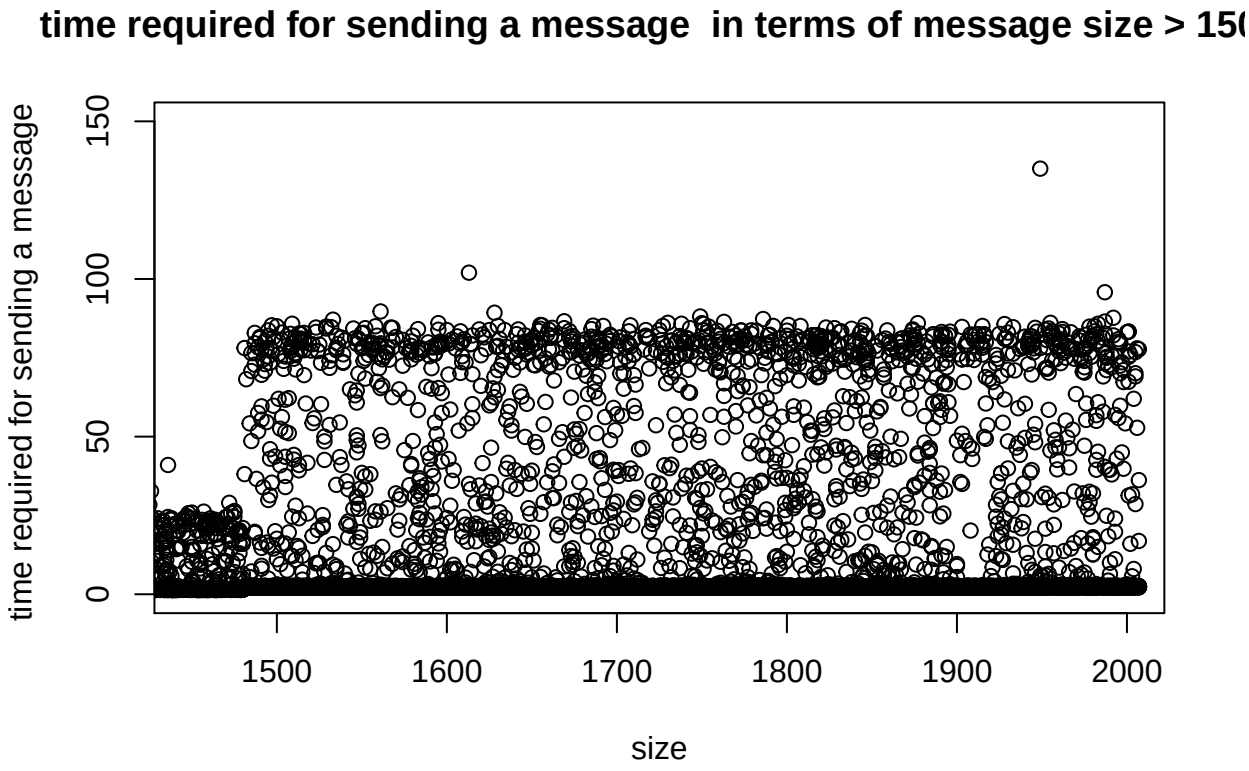
Curve representing the time required for sending a message in terms of size of a message, focued on range 0 –> 1500 of a message size

```
plot(df$size, df$time, main ="time required for sending a message  in terms of message size < 1500"  ,
     xlab="size", ylab="time required for sending a message ", xlim=c(0,1500)  , ylim = c(0 , 100))
```



Curve representing the time required for sending a message in terms of size of a message, focued on range 1500 —> 2000 of a message size

```
plot(df$size, df$time, main ="time required for sending a message  in terms of message size > 1500",
     xlab="size", ylab="time required for sending a message ", xlim=c(1450,2000) , ylim=c( 0, 150))
```



We can observe from here that there is 2 cases that should be taken into account, if message size is less than 1500 where we can see usually the mesasge takes time less than 25 ms while if msg size is greater that 1500 usually it takes time less than 100 ms

we need to classify the data, a column that specifies the size if it is >1500 or less

```
checkingSize = function(size)
  return(size > 1500)
df$type = as.numeric(sapply(df$size , checkingSize))
summary(df)
```

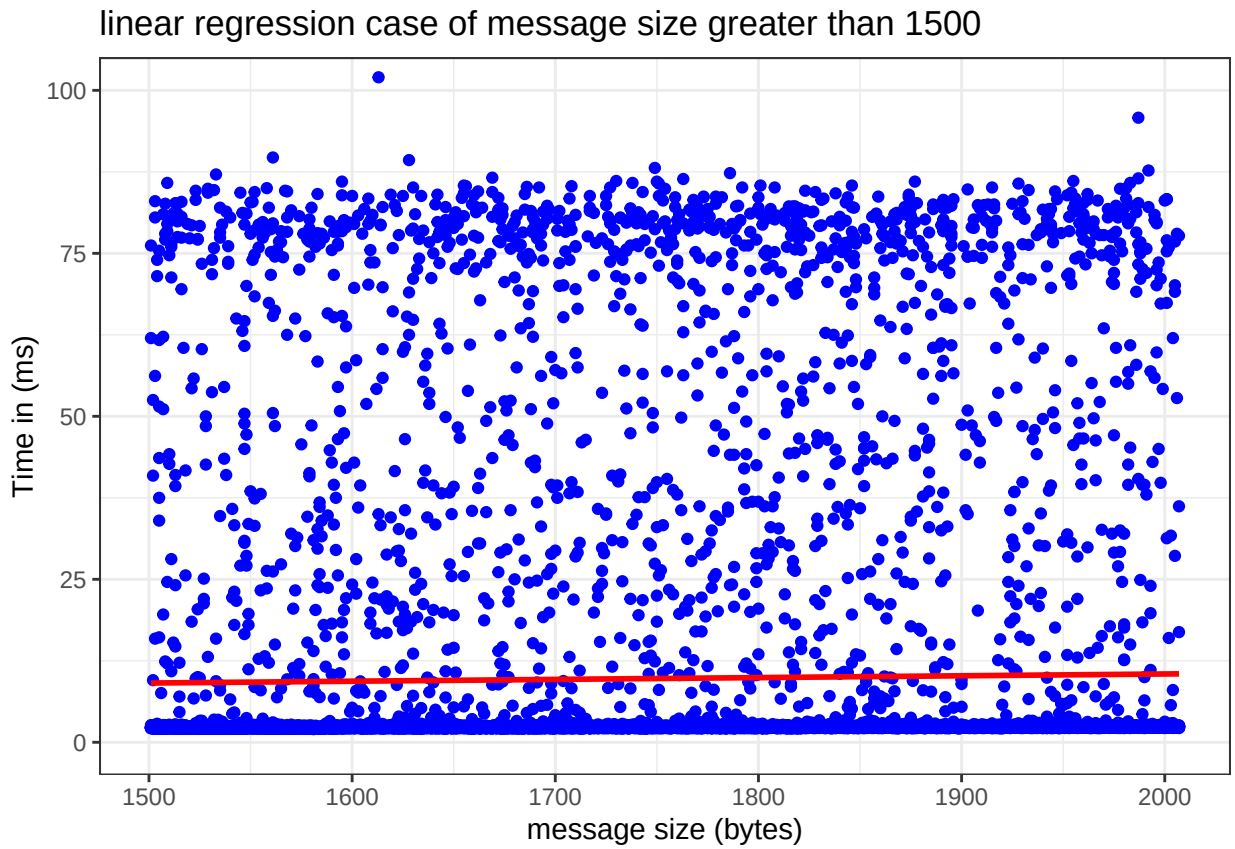
```
##       date                    size                    time
##  Min.   :1421761682.05   Min.   :  24.00000000   Min.   :  1.00000000000
##  1st Qu.:1421764010.13   1st Qu.: 499.00000000   1st Qu.:  1.27000000000
##  Median :1421766410.94   Median : 989.00000000   Median :  1.44000000000
##  Mean   :1421766413.83   Mean   :1000.02572895   Mean   :  5.13666250341
##  3rd Qu.:1421768788.05   3rd Qu.:1499.00000000   3rd Qu.:  2.31000000000
##  Max.   :1421771186.84   Max.   :2007.00000000   Max.   :276.00000000000
##       type
##  Min.   :0.000000000000
##  1st Qu.:0.000000000000
##  Median :0.000000000000
##  Mean   :0.248841856663
##  3rd Qu.:0.000000000000
##  Max.   :1.000000000000
```

drawing the linear regression for messages size greater than 1500

```
data=df[df$type==1,]
reg <- lm(data=df[df$type==1,],time~size)
summary(reg)
```

```
##
## Call:
## lm(formula = time ~ size, data = df[df$type == 1, ])
##
## Residuals:
##              Min               1Q           Median               3Q
##  -8.309137568272  -7.743207113294  -7.356430658944  -6.986271066848
##              Max
## 177.234641609647
##
## Coefficients:
##                       Estimate        Std. Error t value Pr(>|t|)
## (Intercept) 4.920748136138675 2.385178821048836 2.06305 0.039131 *
## size        0.002779466583026 0.001354971167597 2.05131 0.040260 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.75173472908 on 10956 degrees of freedom
## Multiple R-squared:  0.0003839228976947, Adjusted R-squared:  0.0002926837522856
## F-statistic: 4.207874766617 on 1 and 10956 DF,  p-value: 0.04026044412267
```

```
ggplot(data, aes(x=size, y=time))+ ggtitle("linear regression case of message size greater than 1500 ")
  labs(y="Time in (ms)", x = "message size (bytes)") +
  geom_point(color='blue') +
  geom_smooth(color='red' , method = "lm", se = FALSE)+
  theme_bw() +  coord_cartesian(ylim=c(0,100))
```



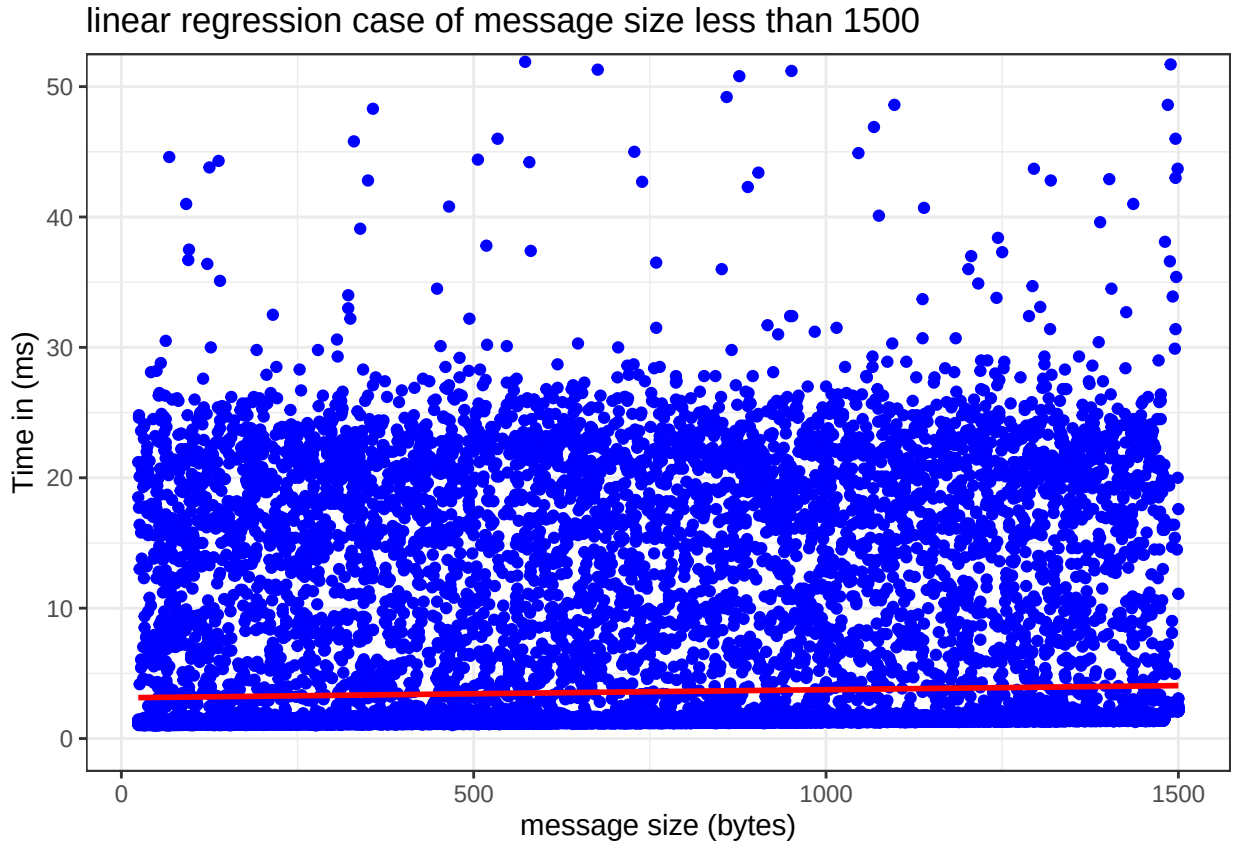
As we can see in the figure $T(S) = L + S/C + \epsilon$ where $L = 4.920748136138675$ $1/C = 0.002779466583026$. The p-value for size is 0.040260 *. A small value means that size is probably an excellent addition to your model One measure very used to test how good the model is the coefficient of determination or R^2. $R^2 = 0.0002926837522856$ which is so small and far from 1. This means that the model explains only 0.2% of the data variability which is really bad. #### drawing the linear regression for messages size less than 1500

```
data=df[df$type==0,]
reg <- lm(data=df[df$type==0,],time~size)
summary(reg)
```

```
##
## Call:
## lm(formula = time ~ size, data = df[df$type == 0, ])
##
## Residuals:
##              Min               1Q           Median               3Q
##  -2.753302112878  -2.379075492067  -2.207241695999  -2.029867784535
##              Max
## 272.425286223605
##
## Coefficients:
##                       Estimate         Std. Error  t value   Pr(>|t|)
## (Intercept) 3.124251115385e+00 7.614339356673e-02 41.03115 < 2.22e-16 ***
## size        6.247748418994e-04 8.834166633007e-05  7.07226 1.5546e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.819329321124 on 33076 degrees of freedom
## Multiple R-squared:  0.001509894750476,  Adjusted R-squared:  0.001479706997868
## F-statistic: 50.01679886878 on 1 and 33076 DF,  p-value: 1.554585337679e-12
```

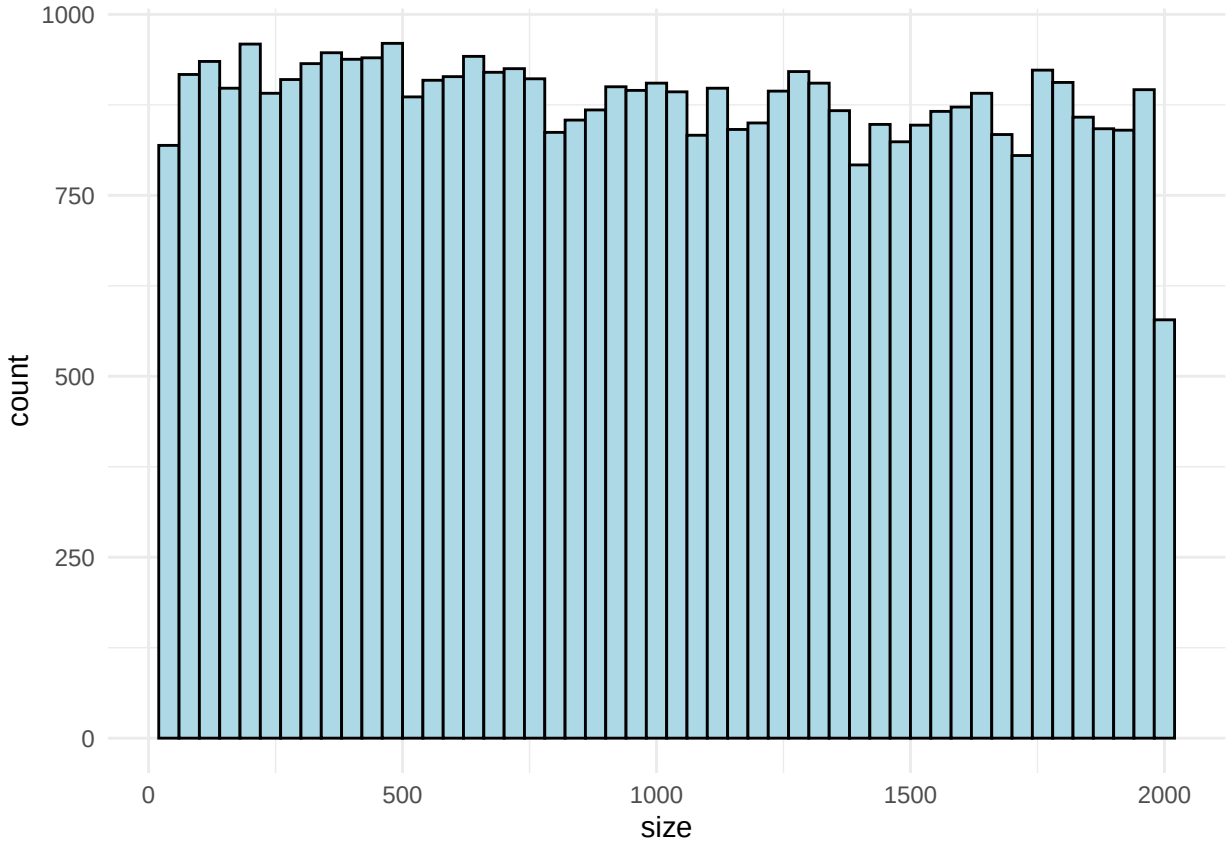
```
ggplot(data, aes(x=size, y=time)) +
  labs(y="Time in (ms)", x = "message size (bytes)") +
  ggtitle("linear regression case of message size less than 1500 ")+
  geom_point(color='blue') +
  geom_smooth(color='red' , method = "lm", se = FALSE)+
  theme_bw() + coord_cartesian(ylim=c(0,50))
```



As we can see in the figure T(S) = L + S/C + ϵ where L = 3.124251115385e+00 1/C = 6.247748418994e-04 Using Pr value we can see that 1.5546e-12 *** is a very small value that means thats probably size is probably an excellent addition to model. Also we can see that R-squared: 0.001479706997868 very low that means also that lm doesn't fit our data very well there is another factor that should be taken into consideration. (size is good but not the only one affecting the time transmission)

Distribution of message sizes

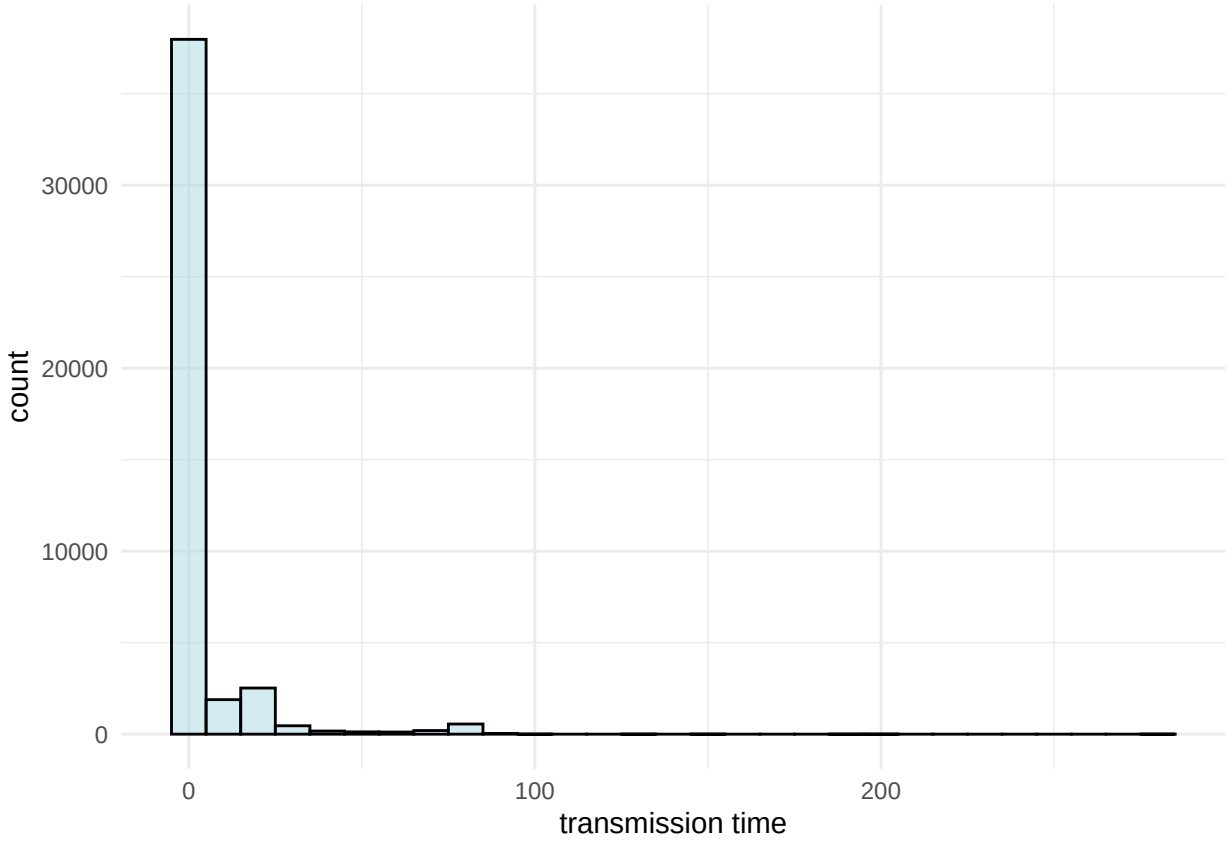
```
ggplot(df[], aes(x=size)) +   geom_histogram(binwidth=40 ,
          color ="black" ,  fill="lightblue", alpha=1)+theme_minimal()
```



in this figure is just for having an overview over how message sizes are distributed.

Distribution of Time

```
ggplot(df, aes(x=time)) +
    labs(y="count", x = "transmission time") +
  geom_histogram(binwidth=10 , color ="black" ,  fill="lightblue", alpha=0.5)+theme_minimal()
```



this figure counts how much time each transmission time happen, where after this result I can see why is the linear regression line at the bottom, because the interval where time transmission is between 0 and 10 have a very high value in terms of others intervals. ## Quantile regression

```
data=df[df$type==0,]
rqfit1 <- rq(time ~ size, tau = 0.25,  data = data )
rqfit2 <- rq(time ~ size, tau = 0.75,  data = data )

summary(rqfit1)
```

```
##
## Call: rq(formula = time ~ size, tau = 0.25, data = data)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value               Std. Error          t value
## (Intercept)    1.09418502202643    0.00096296484671 1136.26683857072521
## size           0.00024229074890    0.00000115781369  209.26574894260335
##             Pr(>|t|)
## (Intercept)    0.00000000000000
## size           0.00000000000000
```

```
summary(rqfit2)
```

```
##
```

```
## Call: rq(formula = time ~ size, tau = 0.75, data = data)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value              Std. Error         t value
## (Intercept)   1.21256290773532   0.00445951251520 271.90481103116906
## size          0.00029822926375   0.00000721448064  41.33759291593831
##             Pr(>|t|)
## (Intercept)   0.00000000000000
## size          0.00000000000000
```

```
anova(rqfit1 , rqfit2)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: time ~ size
## Joint Test of Equality of Slopes: tau in {  0.25 0.75  }
##
##   Df Resid Df  F value     Pr(>F)
## 1  1    66155 65.15658 6.6613e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we can using anaova to see if there is a difference between using conditional quantile with values (0.25 , 0,75) and since 6.6613e-16 *** is extremly small and F value is big that means that there is a difference between these two (slops different)

```
q10 <- seq(0.05, 0.95, by = 0.1)
ggplot(data, aes(x=size, y=time)) +
  labs(y="Time in (ms)", x = "message size (bytes)") +
  ggtitle("quantile regression case of message size less than 1500 ")+
  geom_point(color='blue') +
  geom_quantile( quantiles = q10 , color="red")+

  coord_cartesian(ylim=c(0,2))
```

```
## Smoothing formula not specified. Using: y ~ x
```

Quantile regression has been proposed and used as a way to discover more useful predictive relationships between variables in cases where there is no relationship or only a weak relationship between the means of such variables. the tau option tells rq which conditional quantile we want. We can specify a tau option which tells rq which conditional quantile we want.

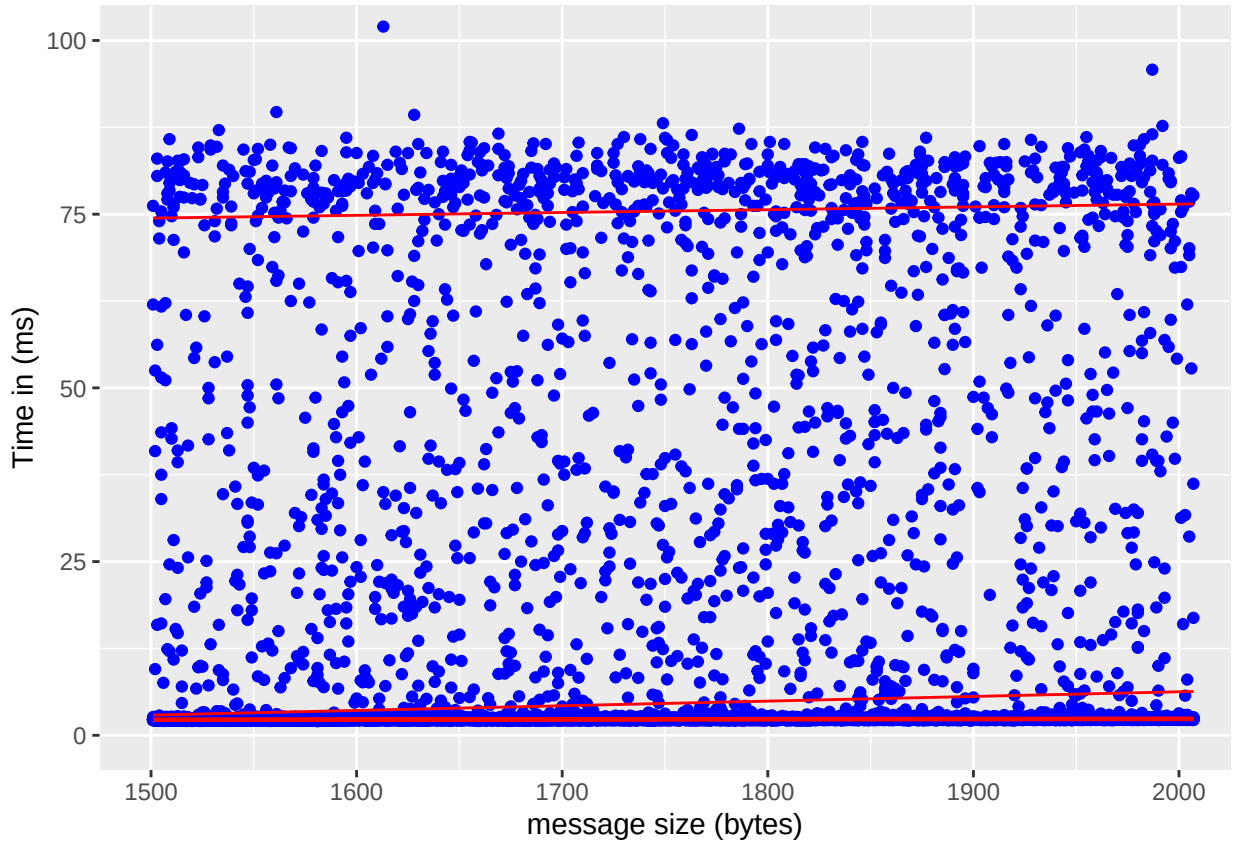
```
data3 =df[df$type==1,]
rqfit1 <- rq(time ~ size, tau = 0.5,  data = data3 )
summary(rqfit1)
```

```
##
## Call: rq(formula = time ~ size, tau = 0.5, data = data3)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value                 Std. Error
## (Intercept) 1.890000000000000e+00 1.533019064410000e-02
## size        2.439024390200000e-04 8.780133869999999e-06
##             t value               Pr(>|t|)
## (Intercept) 1.232861380446989e+02 0.000000000000000e+00
## size        2.777889751056425e+01 0.000000000000000e+00
```

```
q10 <- seq(0.05, 0.95, by = 0.1)
ggplot(data3, aes(x=size, y=time)) +
  labs(y="Time in (ms)", x = "message size (bytes)") +
```

```
  geom_point(color='blue') +
  geom_quantile( quantiles = q10 , color="red")+
  coord_cartesian(ylim=c(0,100) , xlim=c(1500 ,2000))
```

```
## Smoothing formula not specified. Using: y ~ x
```



we can see an interesting results in the figure

Minimum Time Transmission Per Message Size

now it is time to try another than mean regression, i will keep only the minimum time transmission for a message size, i will group by message size then take the minimum between the list of values for this message size.

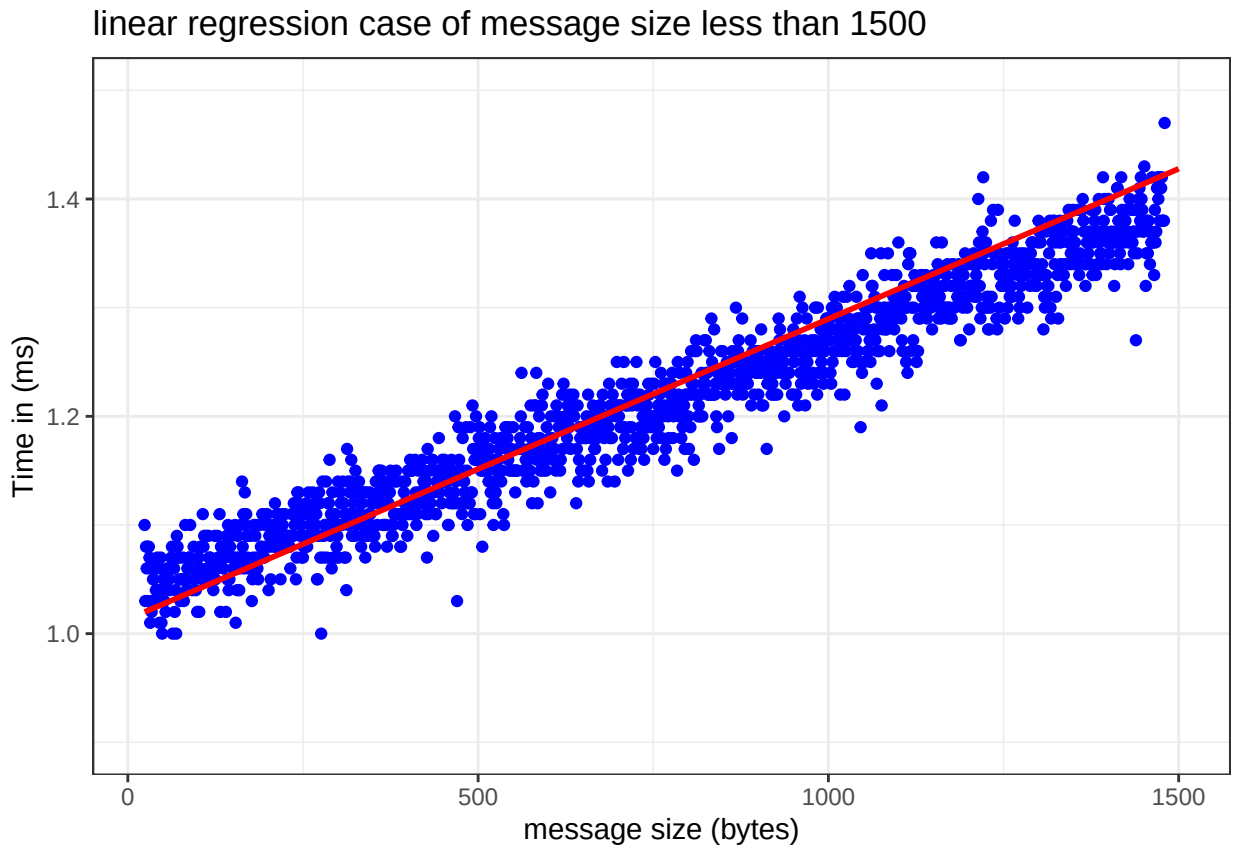
```
#data
data =df
data2 = data %>% group_by(size) %>% slice(which.min(time))



reg <- lm(data=data2[data2$type==0,],time~size)
summary(reg)
```

```
##
## Call:
## lm(formula = time ~ size, data = data2[data2$type == 0, ])
##
## Residuals:
##               Min                1Q            Median                3Q
## -0.14091971707186 -0.02838057418983 -0.00961777092047  0.01211382735773
##               Max
##  0.73498755281643
##
## Coefficients:
##                       Estimate         Std. Error   t value   Pr(>|t|)
## (Intercept) 1.013283665489e+00 4.569442449512e-03 221.75215 < 2.22e-16 ***
## size        2.763280414061e-04 5.233123031470e-06  52.80366 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08575134992889 on 1475 degrees of freedom
## Multiple R-squared:  0.6540179012752,    Adjusted R-squared:  0.6537833371405
## F-statistic: 2788.226350255 on 1 and 1475 DF,  p-value: < 2.2204460493e-16
```

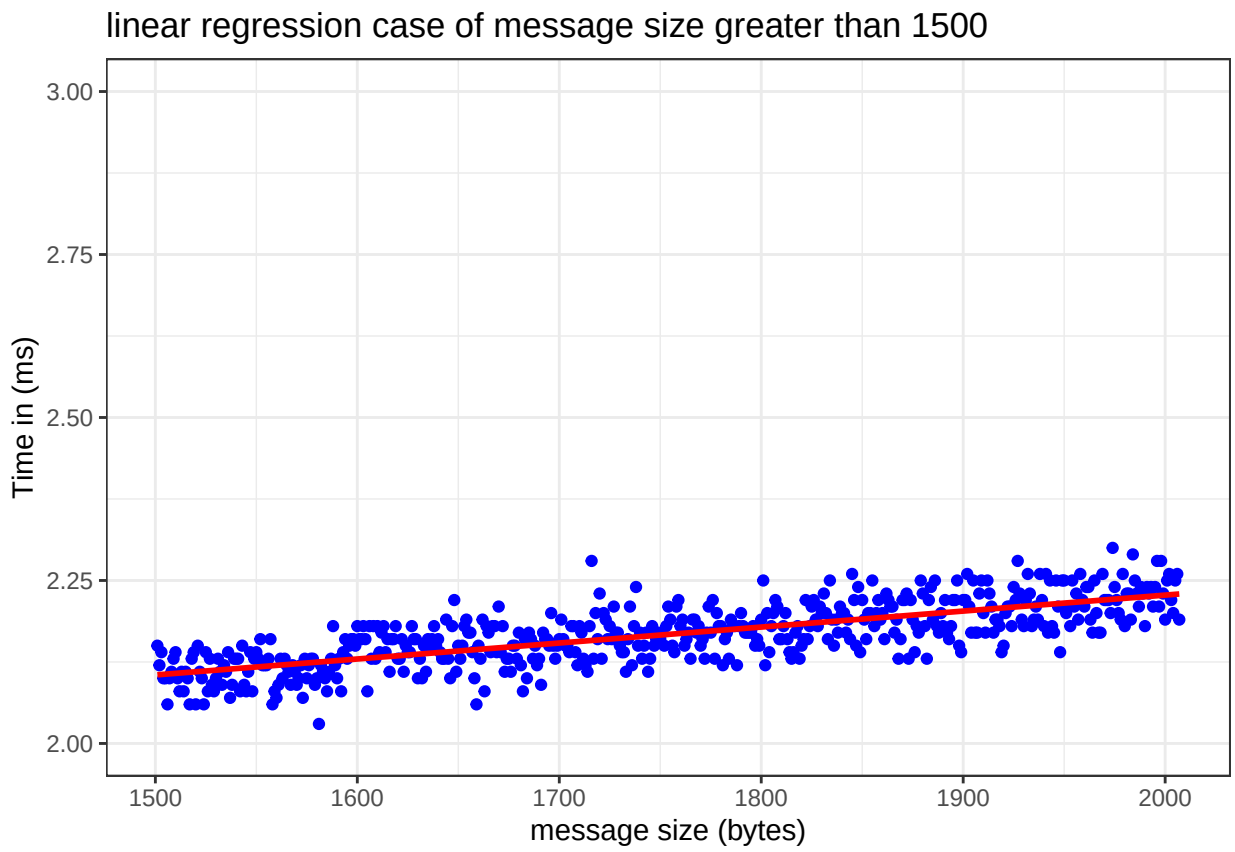
```
ggplot(data2[data2$type==0,], aes(x=size, y=time)) +
  labs(y="Time in (ms)", x = "message size (bytes)") +
  ggtitle("linear regression case of message size less than 1500 ")+
  geom_point(color='blue') +
  geom_smooth(color='red' , method = "lm", se = FALSE)+
  theme_bw() + coord_cartesian(ylim=c(0.9,1.5))
```




```
reg <- lm(data=data2[data2$type==1,],time~size)
summary(reg)
```

```
##
## Call:
## lm(formula = time ~ size, data = data2[data2$type == 1, ])
##
## Residuals:
##                 Min                  1Q              Median
## -0.094921242961429 -0.022297830374753  0.000901497487756
##                  3Q                 Max
##  0.021939269264869  0.122010321773838
##
## Coefficients:
##                       Estimate         Std. Error   t value   Pr(>|t|)
## (Intercept) 1.737653123306e+00 1.671749366114e-02 103.94220 < 2.22e-16 ***
## size        2.449513723314e-04 9.498059803391e-06  25.78962 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03130079519702 on 505 degrees of freedom
## Multiple R-squared:  0.5684146544524,    Adjusted R-squared:  0.5675600300057
## F-statistic:  665.104604361 on 1 and 505 DF,  p-value: < 2.2204460493e-16
```

```r
ggplot(data2[data2$type==1,], aes(x=size, y=time)) +
  labs(y="Time in (ms)", x = "message size (bytes)") +
  ggtitle("linear regression case of message size greater than 1500 ")+
  geom_point(color='blue') +
  geom_smooth(color='red' , method = "lm", se = FALSE)+
  theme_bw() + coord_cartesian(ylim=c(2,3))
```



as we can the results of the linear regression are very good, Adjusted R-squared it is way more better than before, and we can see that message size is an excellent addition to model.