Latency and capacity estimation for a network connection from asymmetric measurements

OMRACHI soukayna

28/12/2019

We use a simple model to evaluate the performance of a network connection while assuming that: T(S) = L + S/C T : time requiered transmitting a message (ms) S : size of the message in bytes L : latency in seconds C : capacity of the network (bytes/second)

In order to evaluate the latency L and the capacity C of this network, we need to analyse a several series of observations of sending time T for different values of S, based on linear regression approach and other approaches.

In this analysis we use two datasets:

1-The first dataset explores a short on-campus connection,  
2-The second dataset measures the performance of a connection to a remote Web site that is popular and therefore has a heavy load.

###### Uploading and cleaning my data frame (dataset 1)

Let’s start with our first dataset (on-campus connection) We need to prepare our dataset and check its integrity

## loading library

library(dplyr) # used to manipulate data and to work with data frame like objects

##   
## 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(lubridate) # make dealing with our dates and times easier

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(tidyr) # make it easy to “tidy” our data  
 # The two most important properties of tidy data are:  
 ## Each column is a variable.  
 ## Each row is an observation.

## loading our data set

First step is to read the data from the file liglab2.log. Second step is to select the columns that we need for our analyse (v1 for date, v2 for size of message, v9 for the time requiered for sending data).

myDataset = read.table('liglab2.log', sep=' ' , na.strings = "" , header=FALSE , fill = TRUE )  
myDataset = myDataset %>% select(V1, V2, V9)   
head(myDataset);

## V1 V2 V9  
## 1 [1421761682.052172] 665 time=22.5  
## 2 [1421761682.277315] 1373 time=21.2  
## 3 [1421761682.502054] 262 time=21.2  
## 4 [1421761682.729257] 1107 time=23.3  
## 5 [1421761682.934648] 1128 time=1.41  
## 6 [1421761683.160397] 489 time=21.9

## remove NA values from our data frame

Our analysis might require that all rows in a table have complete cases (i.e. no missing values). We can use the function(x) any(is.na(x)) to eliminate lines without available data.

line\_NA = apply(myDataset , 1 , function(x) any(is.na(x)))  
myDataset =myDataset %>% drop\_na()

Then, we change the name of our columns

names(myDataset)[1] <- "date";  
names(myDataset)[2] <- "size";  
names(myDataset)[3] <- "time";  
  
head(myDataset);

## date size time  
## 1 [1421761682.052172] 665 time=22.5  
## 2 [1421761682.277315] 1373 time=21.2  
## 3 [1421761682.502054] 262 time=21.2  
## 4 [1421761682.729257] 1107 time=23.3  
## 5 [1421761682.934648] 1128 time=1.41  
## 6 [1421761683.160397] 489 time=21.9

tail(myDataset)

## date size time  
## 44408 [1421771185.805877] 572 time=1.29  
## 44409 [1421771186.011910] 1338 time=1.47  
## 44410 [1421771186.222729] 1515 time=7.02  
## 44411 [1421771186.429007] 1875 time=2.33  
## 44412 [1421771186.634747] 1006 time=1.61  
## 44413 [1421771186.840222] 1273 time=1.35

## convert Time function

We need to convert the time of transmission to double, and to convert the string in the date column into signifant format.

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

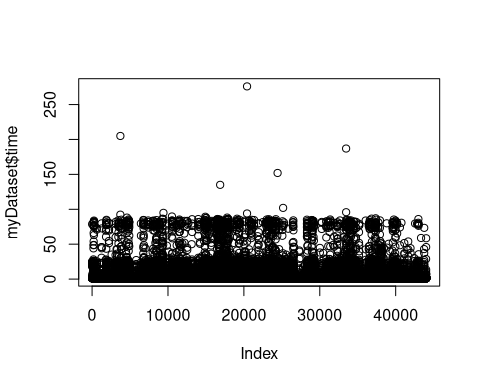
## date size time  
## 1 1421761682.052172 665 22.50  
## 2 1421761682.277315 1373 21.20  
## 3 1421761682.502054 262 21.20  
## 4 1421761682.729257 1107 23.30  
## 5 1421761682.934648 1128 1.41  
## 6 1421761683.160397 489 21.90

We see here that we have our dataset with signifant format of the date, size and time.

###### Plotting data

we need here to plot the time evolution of transmission time

plot(myDataset$time, col="black")

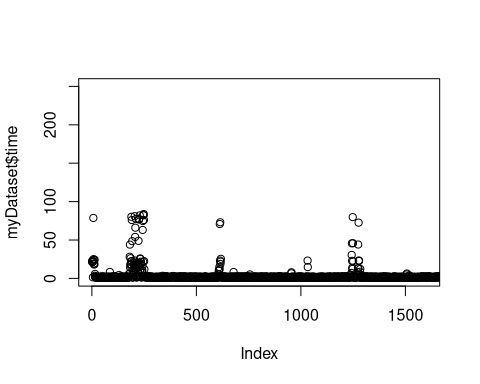


summary(myDataset)

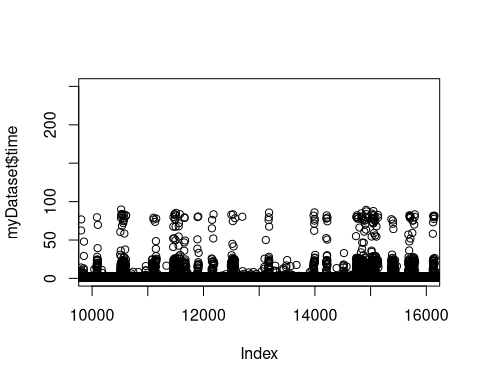
## date size time   
## Min. :1421761682.05 Min. : 24.00000000 Min. : 1.00000000000   
## 1st Qu.:1421764010.13 1st Qu.: 499.00000000 1st Qu.: 1.27000000000   
## Median :1421766410.94 Median : 989.00000000 Median : 1.44000000000   
## Mean :1421766413.83 Mean :1000.02572895 Mean : 5.13666250341   
##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

As we can see it seems that our data is not well explained, so I think we should observe the evolution of the time of transmission for several smaller ranges. Here we plot two samples of our data, one from column 1 to 1600 and the other from 10000 to 16000.

plot(myDataset$time, xlim=c(1,1600), ylim=c(0,250),col="black")

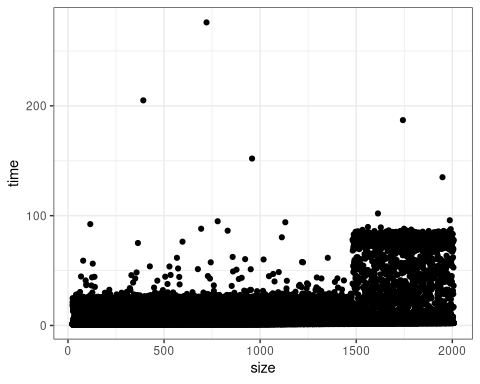


plot(myDataset$time, xlim=c(10000,16000), ylim=c(0,250),col="black")

 We can here observe that the time required for transmitting a message can increase suddenly to 100 ms several times. However, for the rest of the time, the time of transmission is clearly near to the minimum. So, we need to know if those variations could be explained solely by message size?

We need here to plot the transmission time as a function of message size:

library("ggplot2")  
ggplot(data = myDataset, mapping = aes(x = size, y = time)) +   
 theme\_bw() +  
 geom\_point()

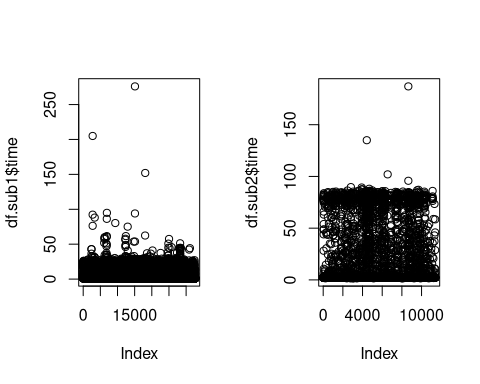


We can observe that we have two main behaviors: - The first one is when message size is less than 1480 - The second behavior is when messgae size is higher than 1480.

To explain this difference, we know that the maximum size of an ethernet packet is 1500 Bytes (about 1480 for data + 20 bytes for headers). So, when packet size exceeds the maximum size allowed, we use the fragmentation, it means that we need to split our data to several packets and send them.In consequence the time required for the transmission of the message when its size exceeds 1480 bytes could increase.

In result, we are gonna to divide our data into two classes. Thr first class is for data when message size is less than 1480 and the other class for data when message size is greater than 1480.

df.sub1= filter(myDataset, size < 1480)  
df.sub2= filter(myDataset, size >= 1480)  
par(mfrow=c(1,2));  
plot(df.sub1$time, col="black")  
plot(df.sub2$time, col="black")



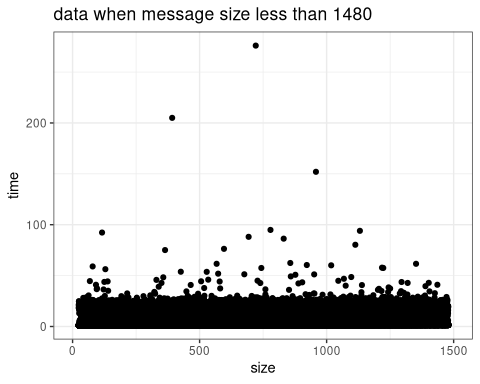
par(mfrow=c(1,1))

###### Analyze: Linear regression

## first dataset (message size less than 1480)

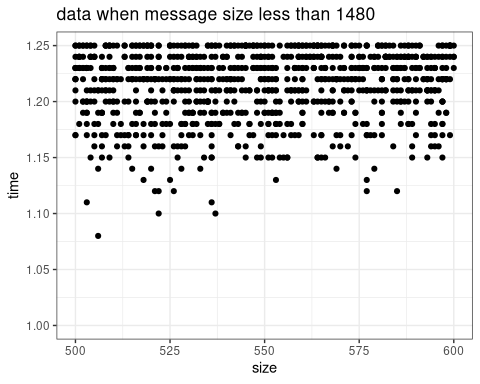
we plot the first dataset, and zoom it to have a better look to our data.

ggplot(data = df.sub1, mapping = aes(x = size, y = time)) +   
 ggtitle("data when message size less than 1480 ")+  
 theme\_bw() +  
 geom\_point() + xlim(0, 1500);

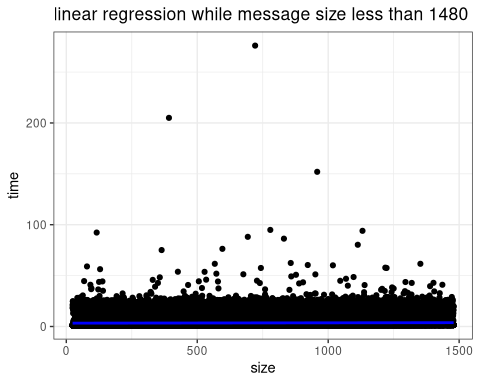


ggplot(data = df.sub1, mapping = aes(x = size, y = time)) +  
 ggtitle("data when message size less than 1480 ")+  
 geom\_point() +  
 theme\_bw() +  
 xlim(500,600) +  
 ylim(1,1.25)

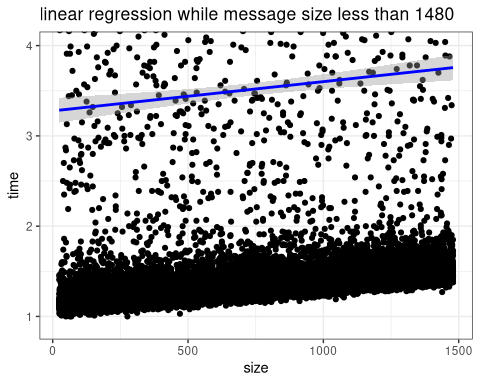
## Warning: Removed 31727 rows containing missing values (geom\_point).

 Here, we plot our dataset with linear regression .

ggplot(data=df.sub1,aes(x=size,y=time)) + theme\_bw() +  
 ggtitle("linear regression while message size less than 1480 ")+  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue")

 we are going to zoom an interval time from 0.9 ms to 4 ms to observe better our data.

ggplot(data=df.sub1,aes(x=size,y=time)) + theme\_bw() +  
 ggtitle("linear regression while message size less than 1480 ")+  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue")+   
 coord\_cartesian(ylim=c(0.9, 4))



we need then to check our model:

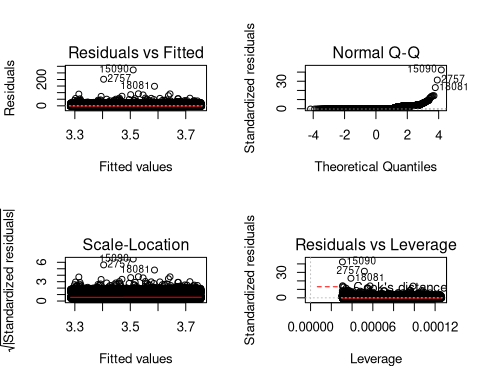
linear\_reg <- lm(time ~ size,data=df.sub1)   
summary(linear\_reg);

##   
## Call:  
## lm(formula = time ~ size, data = df.sub1)  
##   
## Residuals:  
## Min 1Q Median 3Q   
## -2.473136753604 -2.245787590758 -2.188840123602 -2.087867929896   
## Max   
## 272.489541606717   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.276807839256e+00 7.233284317291e-02 45.30180 < 2.22e-16 \*\*\*  
## size 3.240645686923e-04 8.503986599611e-05 3.81074 0.00013881 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 6.431928797604 on 32647 degrees of freedom  
## Multiple R-squared: 0.0004446123804373, Adjusted R-squared: 0.0004139953133984   
## F-statistic: 14.52171691926 on 1 and 32647 DF, p-value: 0.0001388071205714

We use summary to have several informations about our model. - R-squared: the value is very low (near to 0), we can conclude that the fitting of our model is not good. - Degree of freedom: we have here a small R squared, which means that the number of parameters used in our model is not high, so we gain in term of confidence and freedom. - Intercept: 3.276 (-/+) 2*7.23e-02 - capacity = 1/intercept (bytes/s) - Latency = 3.24 (-/+) 2* 8.5e-05 (ms)

We need now to check our hypothesis:

par(mfrow=c(2,2));  
plot(linear\_reg);



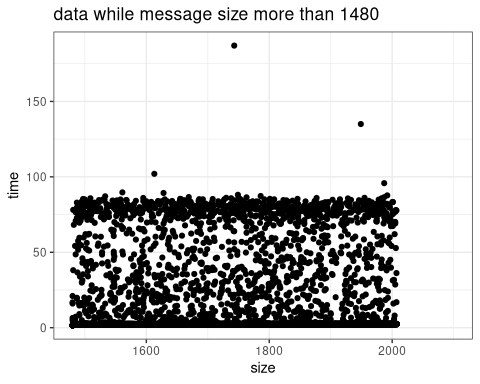
par(mfrow=c(1,1))

* risiduals vs fitted values: what we are looking here is the noise that should be independent,if our model is true we should see something homoscedastic, which is not the case here because we have some wrong patterns.
* normality qq plots: we compare here our samples with a theoretical distribution, for every value we are plotting the position with the respected theoretcal quantile.

## second dataset (message size more than 1480)

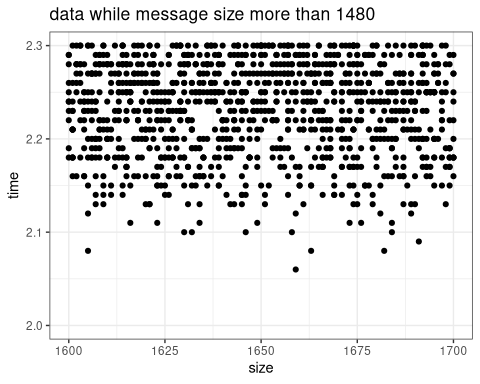
Let’s do the same thing for the second dataset

ggplot(data = df.sub2, mapping = aes(x = size, y = time)) +   
 ggtitle("data while message size more than 1480 ")+  
 theme\_bw() +  
 geom\_point() + xlim(1480, 2100);

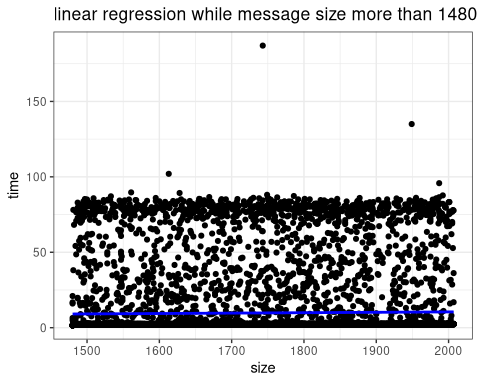


ggplot(data = df.sub2, mapping = aes(x = size, y = time)) +  
 ggtitle("data while message size more than 1480 ")+  
 geom\_point() +  
 theme\_bw() +  
 xlim(1600,1700) +  
 ylim(2,2.3)

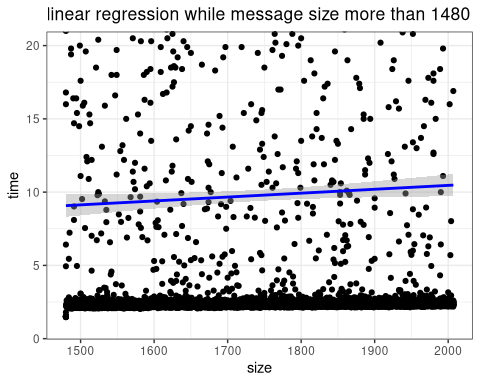
## Warning: Removed 10244 rows containing missing values (geom\_point).



ggplot(data=df.sub2,aes(x=size,y=time)) + theme\_bw() +  
 ggtitle("linear regression while message size more than 1480 ")+  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue")



ggplot(data=df.sub2,aes(x=size,y=time)) + theme\_bw() +  
 ggtitle("linear regression while message size more than 1480 ")+  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue")+  
 coord\_cartesian(ylim=c(0.9, 20))



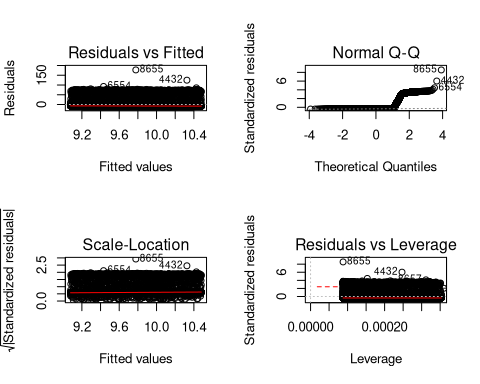
linear\_reg2 <- lm(time ~ size,data=df.sub2)   
summary(linear\_reg2);

##   
## Call:  
## lm(formula = time ~ size, data = df.sub2)  
##   
## Residuals:  
## Min 1Q Median 3Q   
## -8.289576372548 -7.729862195382 -7.343553753453 -6.977367637803   
## Max   
## 177.222149918791   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.144861725886821 2.236852436119508 2.30005 0.021464 \*  
## size 0.002658054133862 0.001277542355703 2.08060 0.037493 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 20.73042860556 on 11385 degrees of freedom  
## Multiple R-squared: 0.0003800833920393, Adjusted R-squared: 0.0002922819061711   
## F-statistic: 4.328894759371 on 1 and 11385 DF, p-value: 0.03749287358138

We check our model. - R-squared: the value is very low (near to 0), we can conclude that the fitting of our model is not good. - Degree of freedom: we have here a small R squared, which means that the number of parameters used in our model is not high, so we gain in term of confidence and freedom. - Intercept: 5.144 (-/+) 2*2.23 - capacity = 1/intercept )(bytes/s) - Latency = 0.0026 (-/+) 2* 0.0012 (ms)

We need now to check our hypothesis:

par(mfrow=c(2,2));  
plot(linear\_reg2);

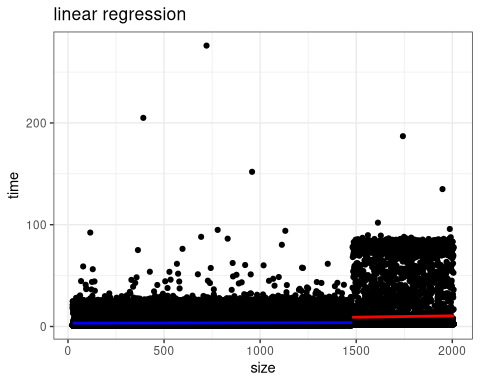


par(mfrow=c(1,1))

similar to what we have said for the first dataset

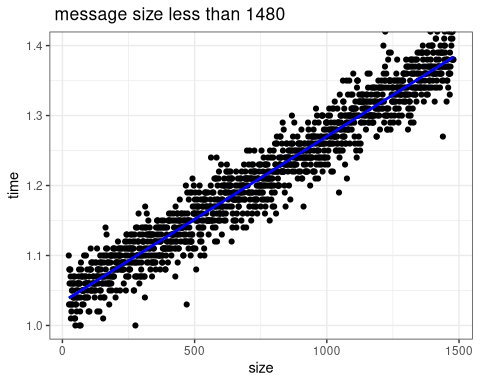
we superpose here linear regression for the two dataset

ggplot(data = myDataset,aes(x=size,y=time)) + theme\_bw() +  
 ggtitle("linear regression ")+  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',data= df.sub1,color="blue")+   
 geom\_smooth(method='lm',data= df.sub2,color="red")

 ###### Analyze: second approach: filtering data while selecting the smallest times per sizes

## first dataset (message size less than 1480)

df.sub1.min = aggregate(time ~ size, df.sub1, min)  
ggplot(data = df.sub1.min, mapping = aes(x = size, y = time)) +  
 ggtitle(" message size less than 1480")+  
 geom\_point(color="black") +  
 theme\_bw() +  
 geom\_smooth(method='lm',data= df.sub1.min,color="blue")+  
 coord\_cartesian(ylim=c(1, 1.4))

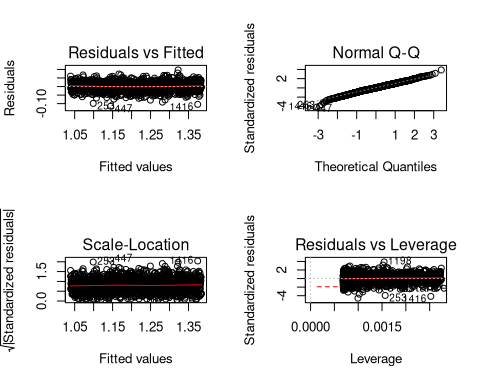
 we check our model

linear.reg.min1 <- lm(time ~ size,data=df.sub1.min)   
summary(linear.reg.min1);

##   
## Call:  
## lm(formula = time ~ size, data = df.sub1.min)  
##   
## Residuals:  
## Min 1Q Median 3Q   
## -0.114832983287612 -0.014727774153971 0.000285265883385 0.016752376201885   
## Max   
## 0.097298368999340   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.033517051963e+00 1.347027683947e-03 767.25747 < 2.22e-16 \*\*\*  
## size 2.368424070746e-04 1.564394486181e-06 151.39558 < 2.22e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02508982392801 on 1454 degrees of freedom  
## Multiple R-squared: 0.9403477903458, Adjusted R-squared: 0.9403067640668   
## F-statistic: 22920.62096424 on 1 and 1454 DF, p-value: < 2.2204460493e-16

The summary of our regression shows that R-squared is close to 1, that means that the fitting of our model is very good.

par(mfrow=c(2,2));  
plot(linear.reg.min1);

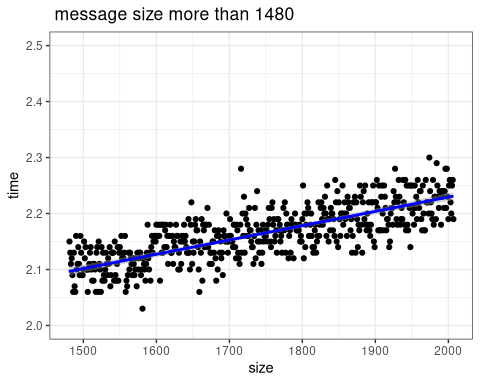


par(mfrow=c(1,1))

* risiduals vs fitted values: what we are looking here is the noise that should be independent,if our model is true we should see something homoscedastic, which is the case here

## second dataset (message size more than 1480)

df.sub2.min = aggregate(time ~ size, df.sub2, min)  
ggplot(data = df.sub2.min, mapping = aes(x = size, y = time)) +  
 ggtitle(" message size more than 1480")+  
 geom\_point(color="black") +  
 theme\_bw() +  
 geom\_smooth(method='lm',data= df.sub2.min,color="blue")+  
 coord\_cartesian(ylim=c(2, 2.5))

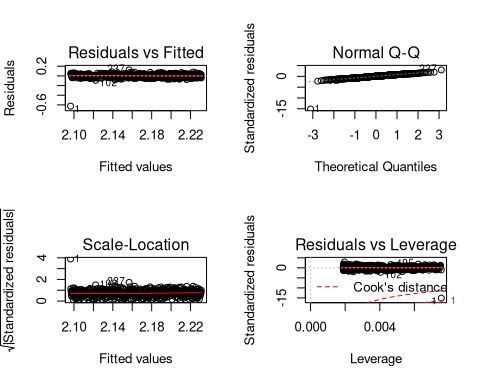


linear.reg.min2 <- lm(time ~ size,data=df.sub2.min)   
summary(linear.reg.min2);

##   
## Call:  
## lm(formula = time ~ size, data = df.sub2.min)  
##   
## Residuals:  
## Min 1Q Median 3Q   
## -0.62649123560779 -0.02137169136193 0.00115740537688 0.02341882008750   
## Max   
## 0.12321852724814   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.718399917922e+00 2.076752321275e-02 82.74458 < 2.22e-16 \*\*\*  
## size 2.554671065439e-04 1.186614068169e-05 21.52908 < 2.22e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.04155937184931 on 526 degrees of freedom  
## Multiple R-squared: 0.4684191121035, Adjusted R-squared: 0.4674085020505   
## F-statistic: 463.5013383221 on 1 and 526 DF, p-value: < 2.2204460493e-16

we have a small R-squared which is too bad because our model doesn’t take into consideration the fragmentation of the packets when the message size exceeds 1480 bytes.

par(mfrow=c(2,2));  
plot(linear.reg.min2);



par(mfrow=c(1,1))

* risiduals vs fitted values: what we are looking here is the noise that should be independent,if our model is true we should see something homoscedastic, which is the case here

###### Analyze: third approach: quantile regression

## first dataset (message size less than 1480)

library(quantreg)

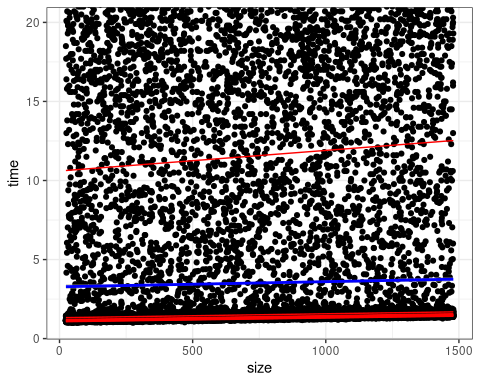
## Loading required package: SparseM

##   
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':  
##   
## backsolve

ggplot(data=df.sub1,aes(x=size,y=time)) + theme\_bw() +  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue") +  
 geom\_quantile(quantiles = 1:9/10 , color="red")+  
 coord\_cartesian(ylim=c(0.9, 20))

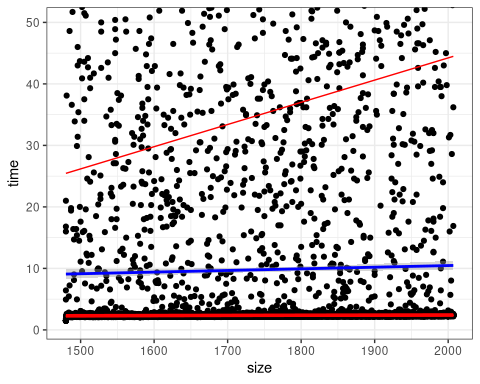
## Smoothing formula not specified. Using: y ~ x



## second dataset (message size more than 1480)

ggplot(data=df.sub2,aes(x=size,y=time)) + theme\_bw() +  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue") +  
 geom\_quantile(quantiles = 1:9/10 , color="red")+  
 coord\_cartesian(ylim=c(0.9, 50))

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



ggplot(data=myDataset,aes(x=size,y=time)) + theme\_bw() +  
 geom\_point(color="black") +  
 geom\_smooth(method='lm',color="blue", data=df.sub1) +  
 geom\_quantile(quantiles = 1:9/10 ,data=df.sub1, color="red")+  
 geom\_smooth(method='lm',color="brown", data=df.sub2) +  
 geom\_quantile(quantiles = 1:9/10 ,data=df.sub2, color="yellow")

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

