Latency and capacity estimation for a network connection from asymmetric measurements

Alexandre D. Jesus

2020-05-30

1 Introduction

A simple and commonly used model for the performance of a network connection is given by

$$T(S) = L + S/C$$

where S denotes the size of a message in bytes, L denotes the latency of the connection in seconds, and C denotes the capacity of the connection. This model $T(S)$ denotes the time required to send a message of size S.

In this work, we are interested in estimating the values L and C for a network connection given sample values of S and $T(S)$.

2 Reading the data

2.1 Dataset format

In this work we consider two datasets for two connections:

- A short on-campus connection: `file://./liglab2.log.gz`
- A connection to `stackoverflow.com`: `file://./stackoverflow.log.gz`

These files contain the raw output of a `ping` command. Each line is of the form

```
[TIMESTAMP] SIZE bytes from DOMAIN (IP): icmp_seq=ICMP_SEQ ttl=TTL time=TIME ms
```

where:

- `TIMESTAMP` denotes the date of the measurement expressed in seconds since January 1, 1970
- `SIZE` denotes the size of the message expressed in bytes, which corresponds to the variable S of our model

- `DOMAIN` and `IP` denote the domain and IP address of the target machine, which should be fixed for each file
- `ICMP_SEQ` and `TTL` are not relevant for our analysis
- `TIME` denotes the round-trip time of the message in milliseconds

An example line looks like this

```
[1421761682.052172] 665 bytes from lig-publig.imag.fr (129.88.11.7): icmp_seq=1
ttl=60 time=22.5 ms
```

2.2 Reading the data

To process these files we devise a function that takes a file as input and returns a dataframe with the relevant data for our analysis

```
read_data <- function(path) {
  if(!file.exists(path)) {
    stop("File does not exist")
  }
  timestamp_r <- "\\[[0-9]+(\\.[0-9]+)?\\]"
  size_r <- "[0-9]+[[:space:]]bytes"
  time_r <- "time=[0-9]+(\\.[0-9]+)?"
  all_r <- sprintf("%s.*%s.*%s", timestamp_r, size_r, time_r)
  lines <- readLines(path)
  ind <- grep(all_r, lines)

  timestamps <- sapply(
    regmatches(lines[ind], regexpr(timestamp_r, lines[ind])),
    function(s) as.numeric(substr(s, 2, nchar(s)-1)),
    USE.NAMES = F
  )

  sizes <- sapply(
    regmatches(lines[ind], regexpr(size_r, lines[ind])),
    function(s) as.numeric(substr(s, 1, nchar(s)-6)),
    USE.NAMES = F
  )

  times <- sapply(
    regmatches(lines[ind], regexpr(time_r, lines[ind])),
    function(s) as.numeric(substr(s, 6, nchar(s))),
    USE.NAMES = F
```

```
  )
  df <- data.frame(
    timestamp = timestamps,
    size = sizes,
    time = times
  )

  df$timestamp <- as.POSIXct(
    df$timestamp,
    origin = "1970-01-01"
  )

  df
}
```

In this function we start by checking if a file exists or not

```
if(!file.exists(path)) {
  stop("File does not exist")
}
```

Then we define regexps that look for the timestamp, size and time data in a line

```
timestamp_r <- "\\[[0-9]+(\\.[0-9]+)?\\]"
size_r <- "[0-9]+[[:space:]]bytes"
time_r <- "time=[0-9]+(\\.[0-9]+)?"
all_r <- sprintf("%s.*%s.*%s", timestamp_r, size_r, time_r)
```

Then, we find all the lines with no missing data, and gather all the data

```
lines <- readLines(path)
ind <- grep(all_r, lines)

timestamps <- sapply(
  regmatches(lines[ind], regexpr(timestamp_r, lines[ind])),
  function(s) as.numeric(substr(s, 2, nchar(s)-1)),
  USE.NAMES = F
)

sizes <- sapply(
  regmatches(lines[ind], regexpr(size_r, lines[ind])),
```

```
  function(s) as.numeric(substr(s, 1, nchar(s)-6)),
  USE.NAMES = F
)

times <- sapply(
  regmatches(lines[ind], regexpr(time_r, lines[ind])),
  function(s) as.numeric(substr(s, 6, nchar(s))),
  USE.NAMES = F
)
```

Finally we aggregate this data into a data frame

```
df <- data.frame(
  timestamp = timestamps,
  size = sizes,
  time = times
)

df$timestamp <- as.POSIXct(
  df$timestamp,
  origin = "1970-01-01"
)

df
```

and quickly check if the function is working correctly

```
head(read_data("./stackoverflow.log.gz"))
```

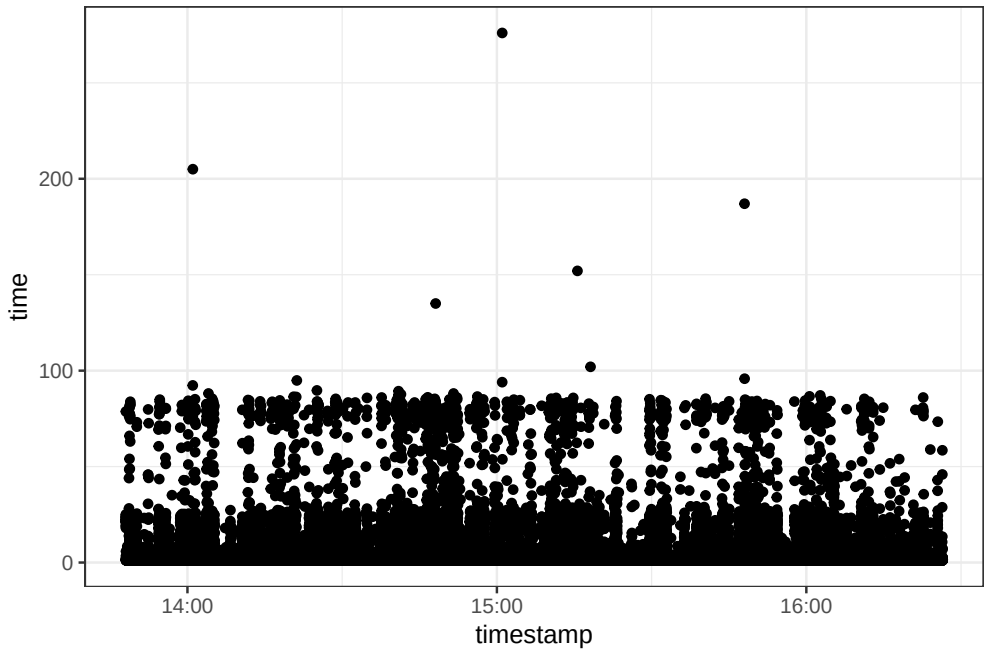
```
            timestamp size time
1 2015-01-20 16:26:43 1257  120
2 2015-01-20 16:26:43  454  120
3 2015-01-20 16:26:43  775  126
4 2015-01-20 16:26:44 1334  112
5 2015-01-20 16:26:44   83  111
6 2015-01-20 16:26:44  694  111
```

3 Analysis of the dataset `liglab2`

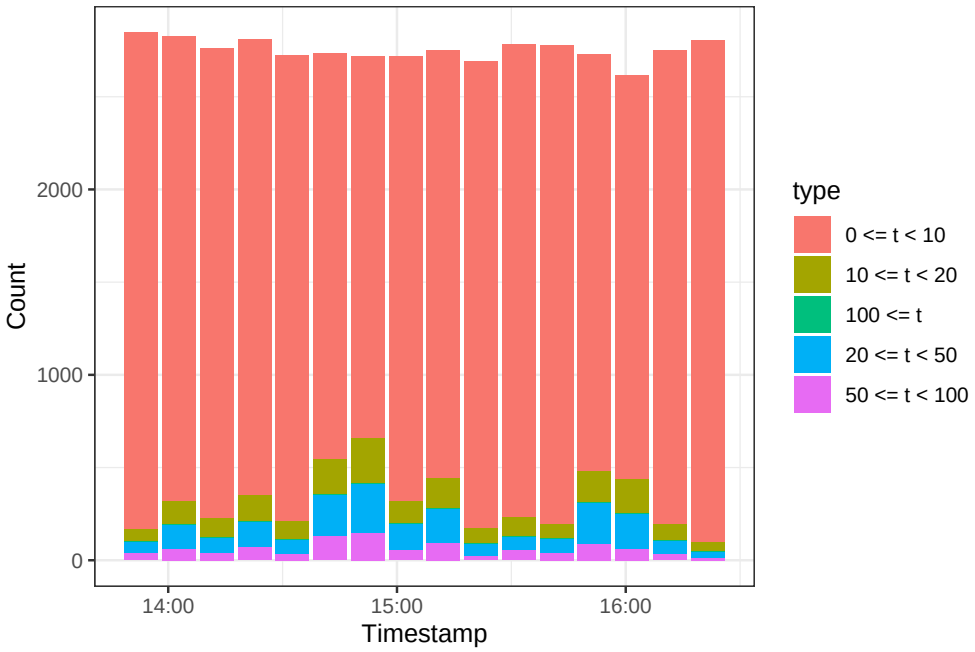
3.1 Effect of timestamp on time

We start by looking at the effect of timestamp into time by plotting all samples from our dataset.

```
1 dat <- read_data("./liglab2.log.gz")
```



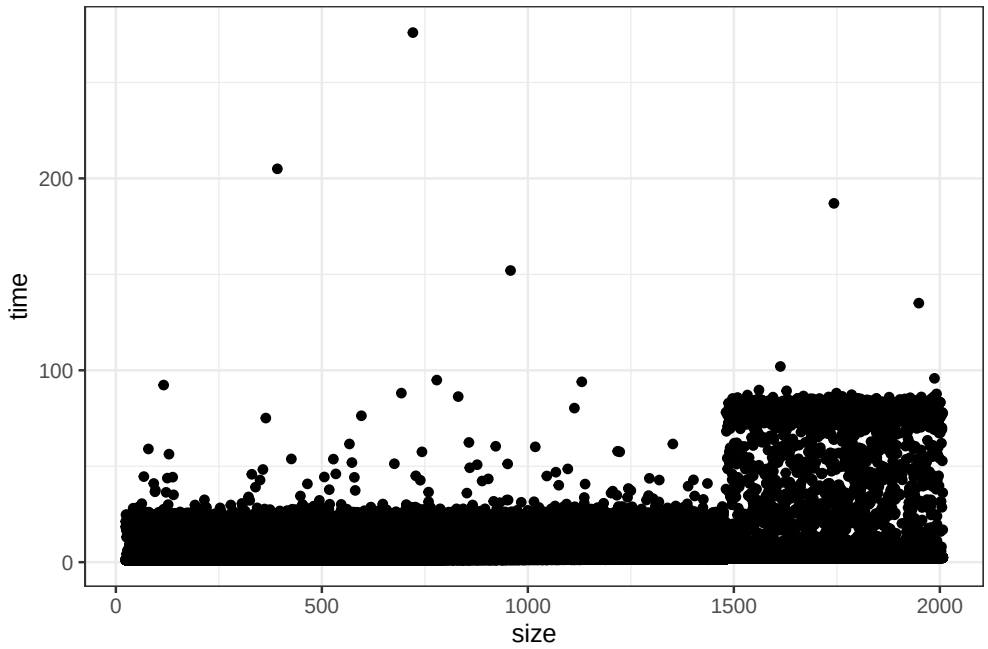
This plot does not show an impact of the timestamp on time. Another way to look at this data is through a bar plot that shows the frequency of time over certain timestamp intervals.



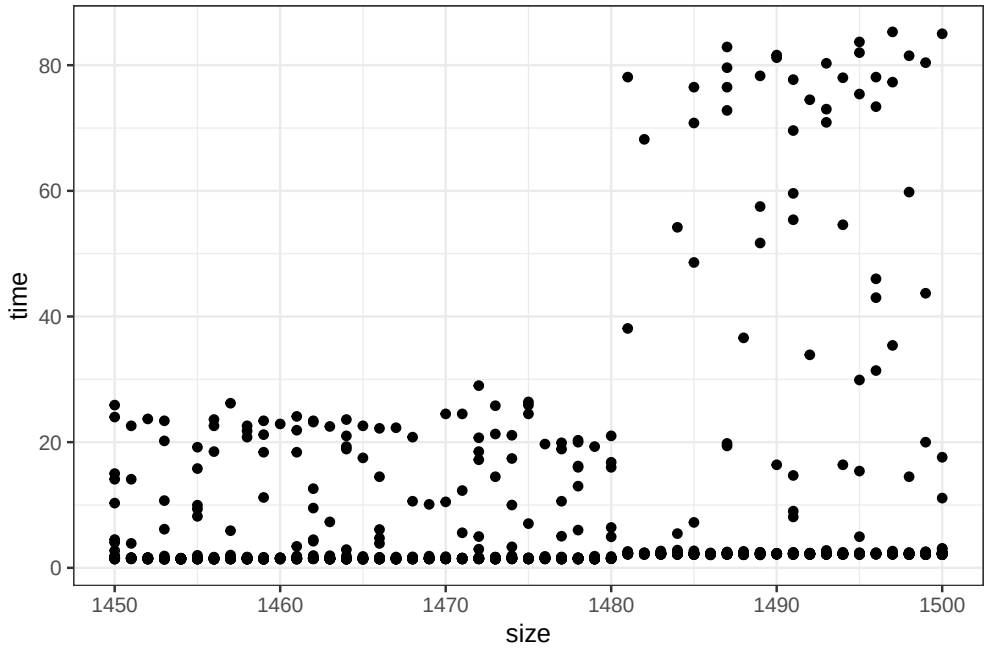
This plot also shows no significant impact of the timestamp on time. There are a couple of intervals where the time seems to increase slightly but nothing too significant. Also we note that the number of samples per timestamp period is consistent.

3.2 Effect of size on time

We start by plotting time over size



This plot shows that there is an increase in time after a certain size. To better determine at what size this increase happens we zoom around that point.



From this plot the increase in time seems to be after size 1480. As such, to be able to deal with these two cases separately, we introduce a class column into the dataset.

```r
dat$class <- NA
dat[dat$size <= 1480,]$class <- "small"
dat[dat$size > 1480,]$class <- "large"
dat$class <- as.factor(dat$class)
```

3.3 Estimating values of L and C

We will estimate the values of L and C for each of the two datasets previously defined.

3.3.1 Linear regression

To estimate the parameters of our network model we start by considering two linear regression models (one for each class). For the class "small" we have the linear regression model

```r
lmMod.small <- lm(time ~ size, data = dat[dat$class == "small",])
lmMod.small
```

```
Call:
lm(formula = time ~ size, data = dat[dat$class == "small", ])

Coefficients:
(Intercept)         size
  3.2756742    0.0003263
```

From this model, we can estimate the latency and capacity values for the network model. In particular the linear regression model gives us the equation

$$T(S) = 3.2756742 + 0.0003263 \times S$$

Then, since our network model is given by

$$T(S) = L + S/C$$

we can estimate a latency $L = 3.2756742$ and a capacity $C = 1/0.0003263 = 3064.664$.

For class "large" we have the linear regression model

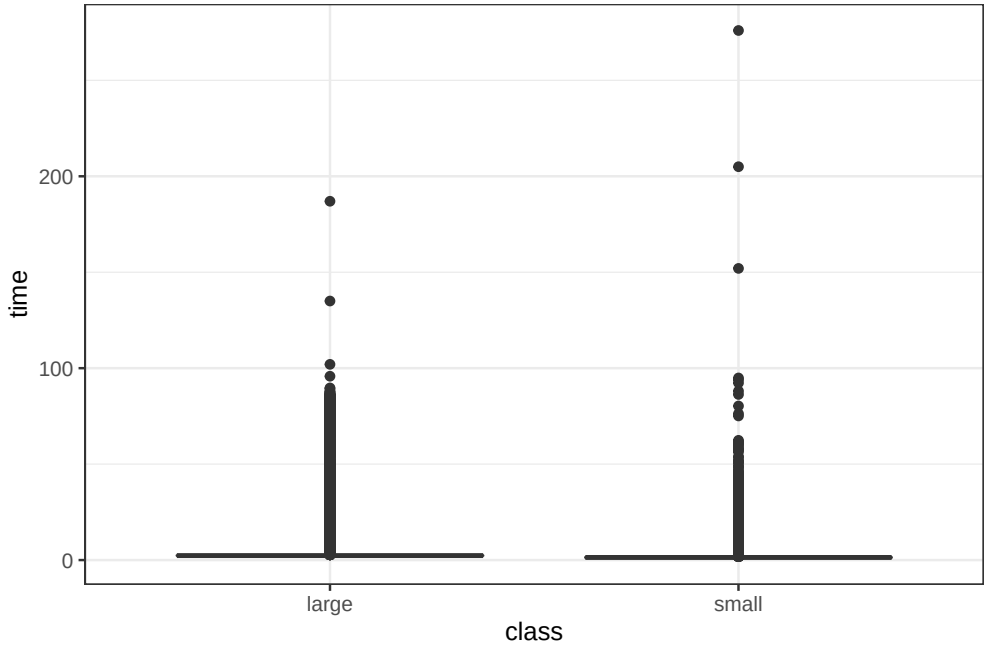
```r
lmMod.large <- lm(time ~ size, data = dat[dat$class == "large",])
lmMod.large
```

```
Call:
```

```
lm(formula = time ~ size, data = dat[dat$class == "large", ])

Coefficients:
(Intercept)         size
   5.289833     0.002579
```

From this we estimate a latency $L = 5.289833$ and a capacity $C = 1/0.002579 = 387.7472$.

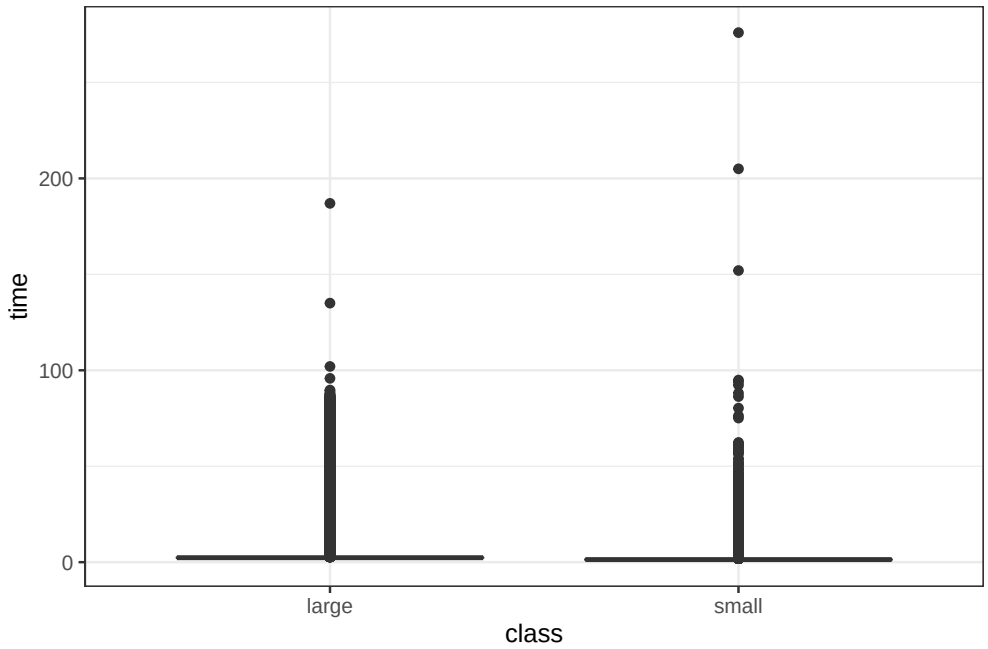
Finally, we plot both the regression models against our samples



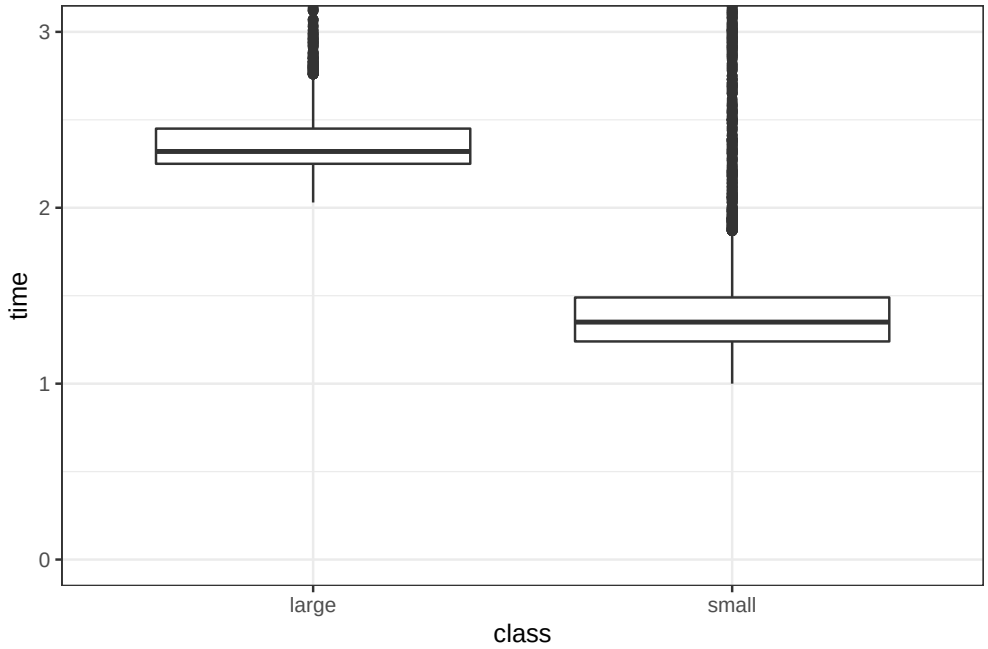
Note that these seem to be overestimating most of the data. In the next section we will look into how we can improve this.

3.3.2 Quantile regression

If we look at a box plot of our data



it indicates that a first look at the points means that we mostly see outliers. We can zoom in further to take a better look at the box plot.



As a result of this asymmetry in the data, its probably better to use quantile regression with respect to the median.

```r
library(quantreg)
```

```r
rqMod.small <- rq(time ~ size, tau = 0.5, data = dat[dat$class == "small",])
rqMod.small
```

```
Call:
rq(formula = time ~ size, tau = 0.5, data = dat[dat$class ==
    "small", ])

Coefficients:
 (Intercept)         size
1.1390594059 0.0002475248

Degrees of freedom: 32667 total; 32665 residual
```

For the "small" class we have an estimated latency $L = 1.139$ and capacity $C = 1/0.0002475248 = 4039.999$.

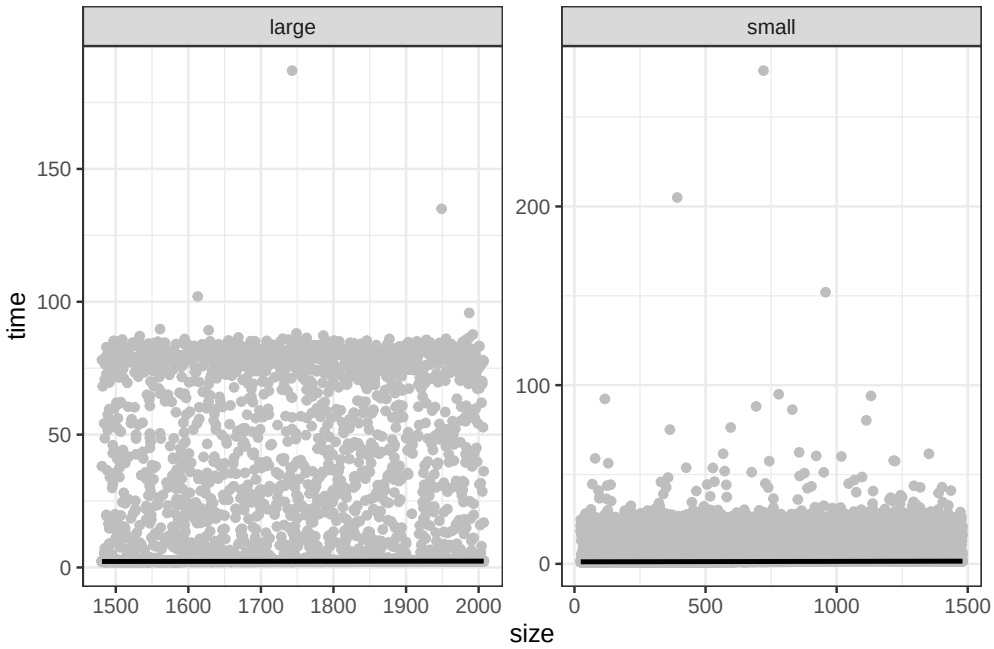
```r
rqMod.large <- rq(time ~ size, tau = 0.5, data = dat[dat$class == "large",])
rqMod.large
```

```
Call:
rq(formula = time ~ size, tau = 0.5, data = dat[dat$class ==
    "large", ])

Coefficients:
 (Intercept)         size
1.8853521127 0.0002464789

Degrees of freedom: 11369 total; 11367 residual
```

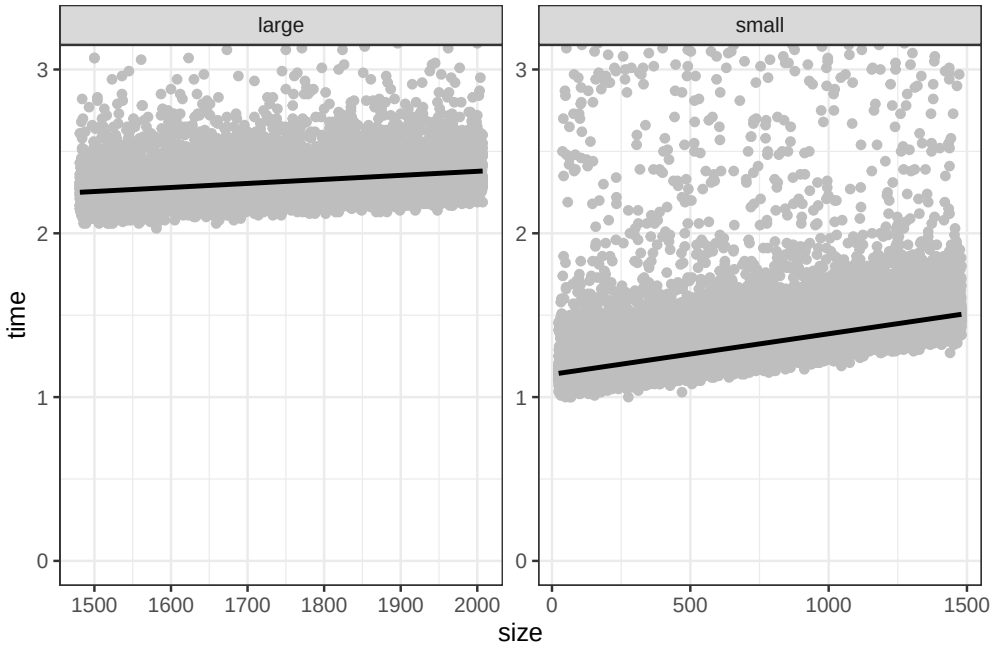
For the "large" class we have an estimated latency $L = 1.885$ and capacity $C = 1/0.0002464789 = 4057.142$.

We can see that the quantile regression gives a different estimate, and that the latency estimate is significantly lower as expected.

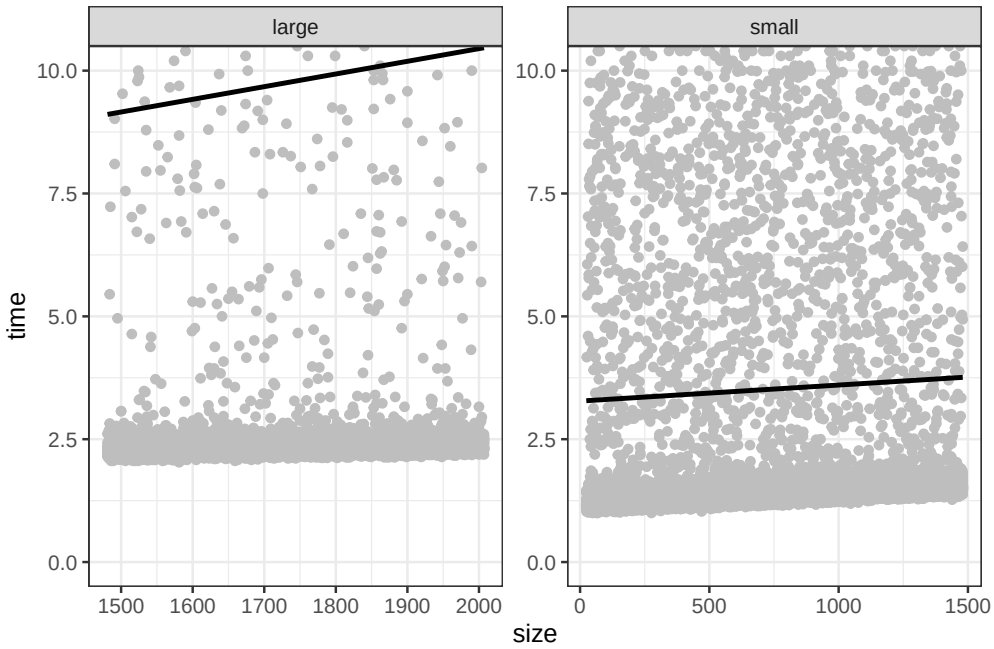
We can plot our models



And zoom in to take a better look



Indeed there seems to be a better fit than with the linear regression which we plot below on a similar scale.



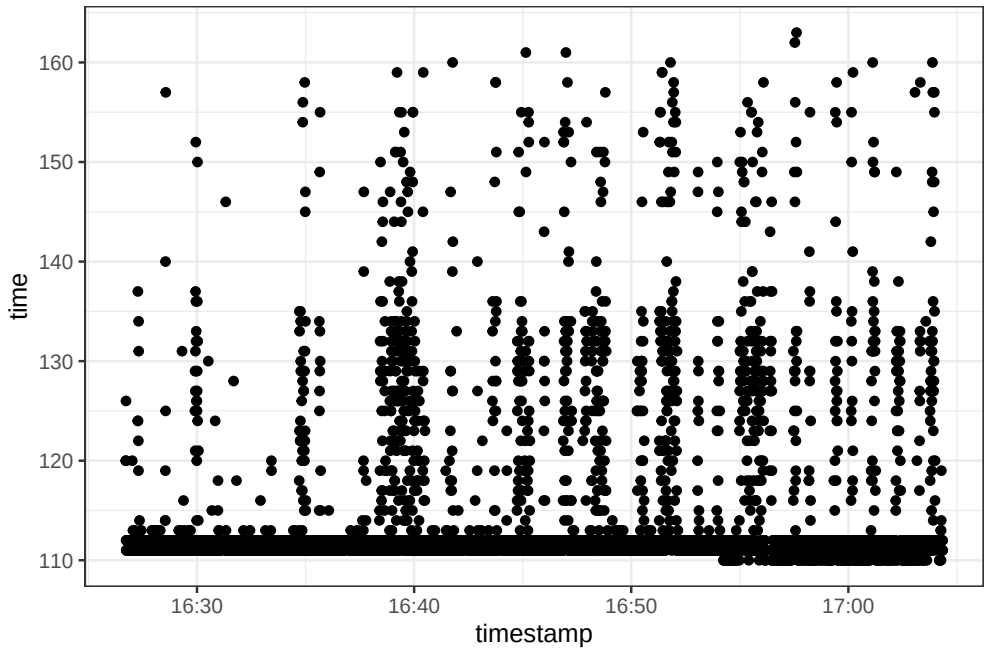
4 Analysis of the dataset `stackoverflow`

Now we will perform a similar analysis on the `stackoverflow` dataset.

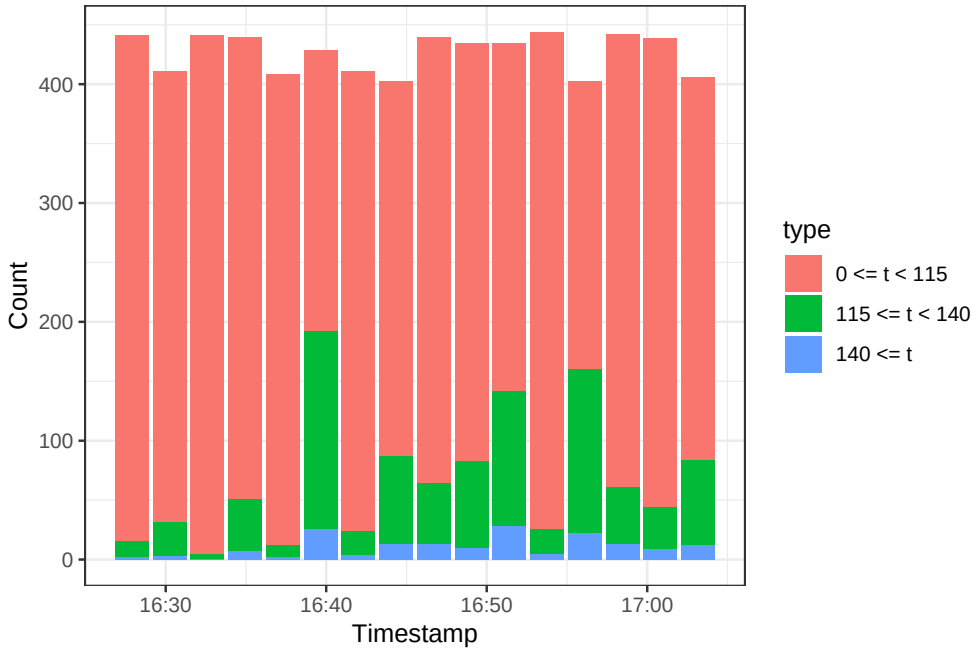
4.1 Effect of timestamp on time

We start by looking of the effect of the timestamp into the time.

```
1 dat <- read_data("./stackoverflow.log.gz")
```



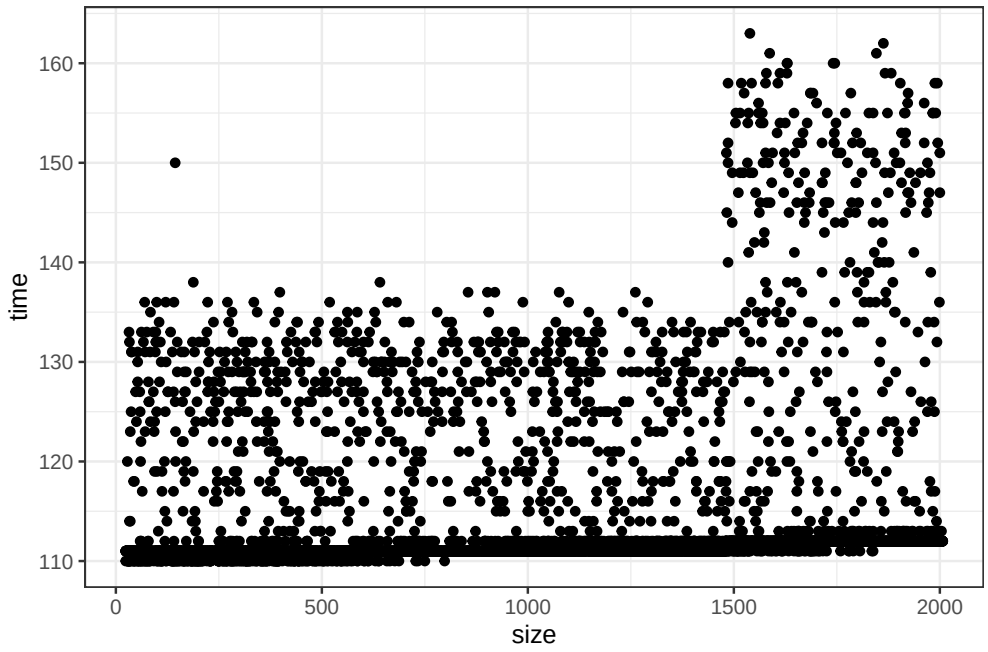
Like in the previous dataset, there seems to be no significant impact of timestamp on time. We can also look frequency bar plots.



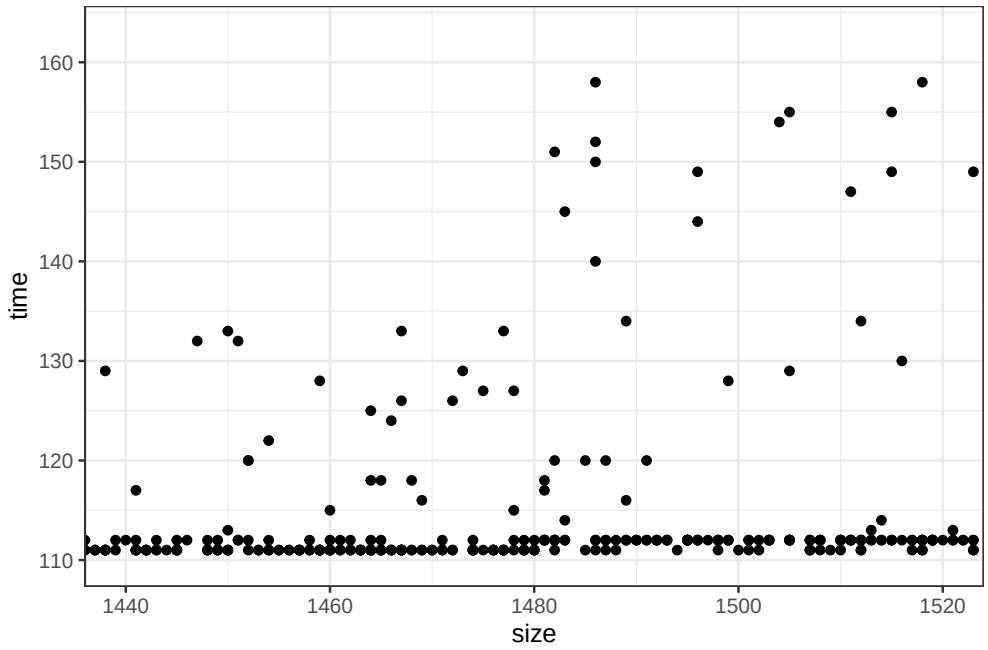
This plot shows that there are a couple of intervals with a slight increase in time (e.g. around 16:40). However, this does not seem to warrant a separation of the data.

4.2 Effect of size on time

We start by plotting the points of time over size



This plot shows that the size seems to have an impact on time. If we zoom in



we see that this difference is after size 1480. As such we split the data into two classes.

```r
dat$class <- NA
dat[dat$size <= 1480,]$class <- "small"
dat[dat$size > 1480,]$class <- "large"
dat$class <- as.factor(dat$class)
```

4.3 Estimating values of L and C

4.3.1 Linear regression

We again start by estimating the parameters of our network model with two linear regression models (one for each class).

```r
lmMod.small <- lm(time ~ size, data = dat[dat$class == "small",])
lmMod.small
```

```
Call:
lm(formula = time ~ size, data = dat[dat$class == "small", ])

Coefficients:
(Intercept)         size
  1.132e+02    4.521e-05
```

For the class "small" (size less than or equal to 1480) the linear model gives us the equation

$$T(S) = 113.2 + 0.00004521 \times S$$

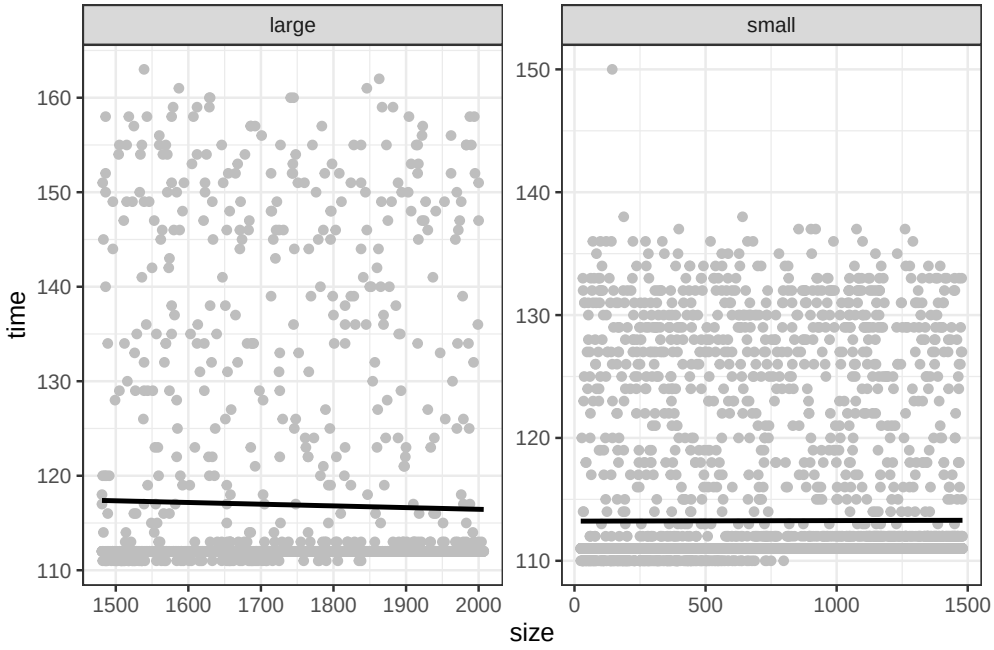
meaning that we can estimate a latency $L = 113.2$ and capacity $C = 1/0.00004521 = 22119$.

```r
lmMod.large <- lm(time ~ size, data = dat[dat$class == "large",])
lmMod.large
```

```
Call:
lm(formula = time ~ size, data = dat[dat$class == "large", ])

Coefficients:
(Intercept)         size
 120.053588    -0.001803
```

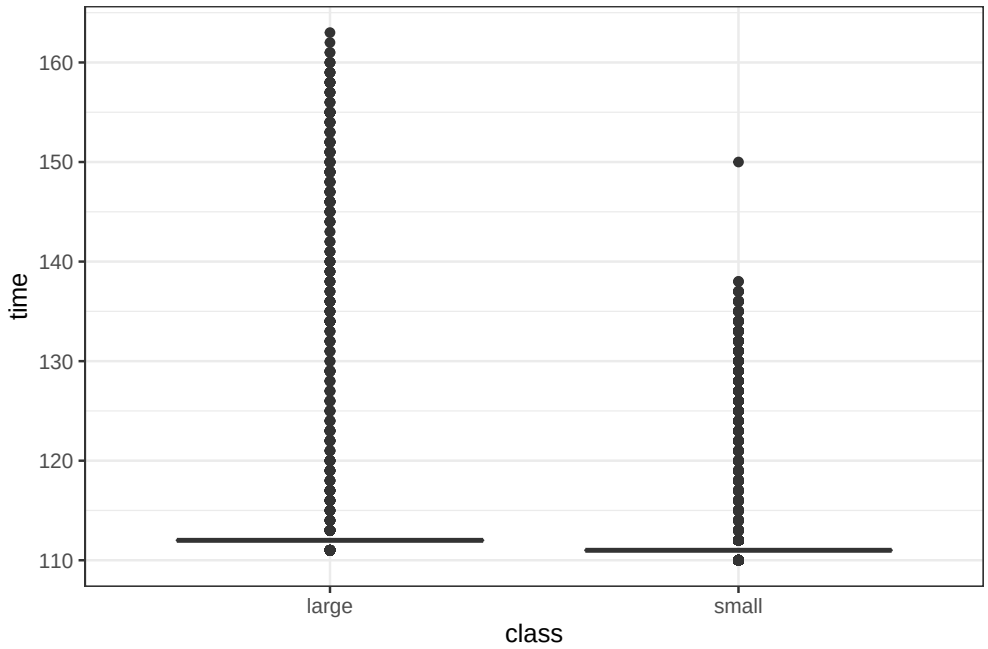
For class "large" we estimate a latency $L = 120.053588$ and capacity $C = 1/-0.001803 = -554.63$. Note that a negative capacity does not make sense in terms of the network model. However, we may assume that this is an artifact of our dataset.

A plot of the regression model against the data seems to indicate that both the models seem to be overestimating the latency due to outliers. The same as it happened for the previous dataset.



4.3.2 Quantile regression

As in the previous case we notice an asymmetry in the data, and the linear regression model seems to influenced by the outliers.



As such, we will once again look into quantile regression with respect to the median.

```
library(quantreg)
```

```
rqMod.small <- rq(time ~ size, tau = 0.5, data = dat[dat$class == "small",])
rqMod.small
```

```
Call:
rq(formula = time ~ size, tau = 0.5, data = dat[dat$class ==
    "small", ])

Coefficients:
 (Intercept)         size
1.110000e+02 7.013151e-18

Degrees of freedom: 5015 total; 5013 residual
```

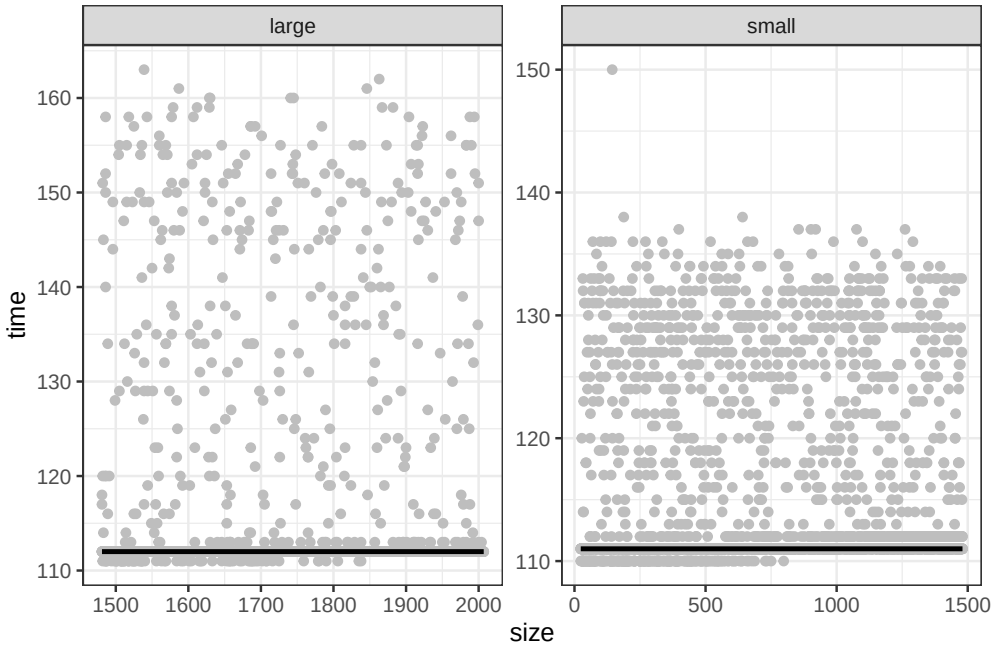
For the "small" class we estimate a latency $L = 111$ and a capacity $C = 1/7.013151e - 18 = 1.425893e + 17$. Note that the capacity seems to have no impact.

```
rqMod.large <- rq(time ~ size, tau = 0.5, data = dat[dat$class == "large",])
```

```
Call:
rq(formula = time ~ size, tau = 0.5, data = dat[dat$class ==
    "large", ])

Coefficients:
  (Intercept)          size
 1.120000e+02 -4.405647e-19

Degrees of freedom: 1809 total; 1807 residual
```

In the "large" class we estimate a latency $L = 112$ and a capacity $C = 1/-4.405647e-19 = -2.269814e+18$. Note that once again we have a negative capacity but that the slope is very close to zero. As such, this seems to be an artifact of the data, and we could change the sign of the slope to be positive with no significant detriment to our model.

Finally, we plot the model along with the data



This plot indicates that similarly to the previous dataset, the quantile model is closer to the actual data.