Estimation de la latence et de la capacité d'une connexion à partir de mesures asymétriques

Louis Le Nézet

Table des matières

Problématique

Un modèle simple et fréquemment utilisé pour décrire la performance d'une connexion de réseau consiste à supposer que le temps d'envoi T pour un message dépend principalement de sa taille S (nombre d'octets) et de deux grandeurs propres à la connexion : la latence L (en secondes) et la capacité C (en octets/seconde). La relation entre ces quatre quantités est T(S) = L + S/C. Ce modèle néglige un grand nombre de détails. D'une part, L et C dépendent bien sûr du protocole de communication choisi mais aussi dans une certaine mesure de S. D'autre part, la mesure de T(S) comporte en général une forte composante aléatoire. Nous nous intéressons ici au temps moyen qu'il faut pour envoyer un message d'une taille donnée.

Objectif

Nous souhaitons ici estimer L et C à partir d'une série d'observations de T pour des valeurs différentes de S.

Préparation des données

1) Le premier jeu de données examine une connexion courte à l'intérieur d'un campus disponible ici L'URL est :

```
data_url1 = "http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/ligl
```

Pour nous protéger contre une éventuelle disparition ou modification du serveur du site, nous faisons une copie locale de ce jeux de données que nous préservons avec notre analyse. Il est inutile et même risquée de télécharger les données à chaque exécution, car dans le cas d'une panne nous pourrions remplacer nos données par un fichier défectueux. Pour cette raison, nous téléchargeons les données seulement si la copie locale n'existe pas.

```
data_file = "liglab2.log.gz"
if (!file.exists(data_file)) {
    download.file(data_url1, data_file, method="auto")
}
```

Il faut ensuite charger les données dans un objet R

```
liglab2 <- read.table(data_file, header=FALSE, sep=";")
```

Il va falloir désormais séparer les données en fonction de leur emplacement. Les données sont pour l'instant en format string. Au début, entre crochet, vous trouvez la date à laquelle la mesure a été prise, exprimée en secondes depuis le 1er janvier 1970. La taille du message en octets est donnée juste après, suivie par le nom de la machine cible et son adresse IP, qui sont normalement identiques pour toutes les lignes à l'intérieur d'un jeu de données. À la fin de la ligne, nous trouvons le temps d'envoi (aller-retour) en millisecondes. Les autres indications, icmp_seq et ttl, n'ont pas d'importance pour notre analyse. Attention, il peut arriver qu'une ligne soit incomplète et il faut donc vérifier chaque ligne avant d'en extraire des informations!

Ici est écris la fonction qui sépare tout d'abord les données. Puis extrait les données nécessaire en les mettant dans une liste

```
require(stringr)
```

```
## Le chargement a nécessité le package : stringr
```

```
extract_infos <- function(x){
infos = str_split(x," ")
infos = str_remove_all(infos[[1]],"\\[|\\]|\\(|\\)|:")
data=list()
data["date_ms"] = as.numeric(infos[1])
data["size"] = as.integer(infos[2])
data["ip"]   = infos[6]
if (any(str_detect(infos,"time="))){
  data["time"] = as.integer(str_remove(infos[str_detect(infos,"time=")],"time="))
}else{
  data["time"] = NA
}

return(data)
}
```

We need now to apply this function on the whole data and use it as a dataframe

```
library(data.table)
df = rbindlist(lapply(liglab2$V1, extract_infos))
```

Pour la gestion de date nous allons utiliser la librairie parsedate

```
df$date=as.POSIXct(df$date_ms,origin='1970-01-01 00:00')
summary(df)
```

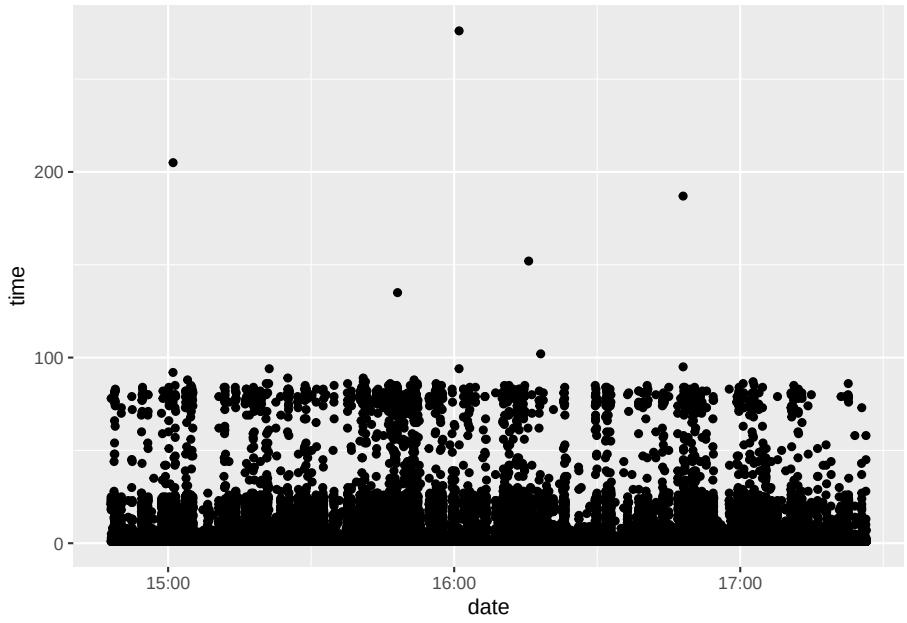
```
##     date_ms               size             ip                 time
##  Min.   :1.422e+09   Min.   :   8.0   Length:44413       Min.   :  1.000
##  1st Qu.:1.422e+09   1st Qu.: 487.0   Class :character   1st Qu.:  1.000
##  Median :1.422e+09   Median : 982.0   Mode  :character   Median :  1.000
##  Mean   :1.422e+09   Mean   : 991.7                      Mean   :  4.794
##  3rd Qu.:1.422e+09   3rd Qu.:1494.0                      3rd Qu.:  2.000
##  Max.   :1.422e+09   Max.   :2007.0                      Max.   :276.000
##                                                          NA's   :377
##       date
##  Min.   :2015-01-20 14:48:02
##  1st Qu.:2015-01-20 15:26:50
##  Median :2015-01-20 16:06:49
##  Mean   :2015-01-20 16:06:53
##  3rd Qu.:2015-01-20 16:46:26
##  Max.   :2015-01-20 17:26:26
##
```

Analyse des données

Evolution du temps de transmission au cours du temps

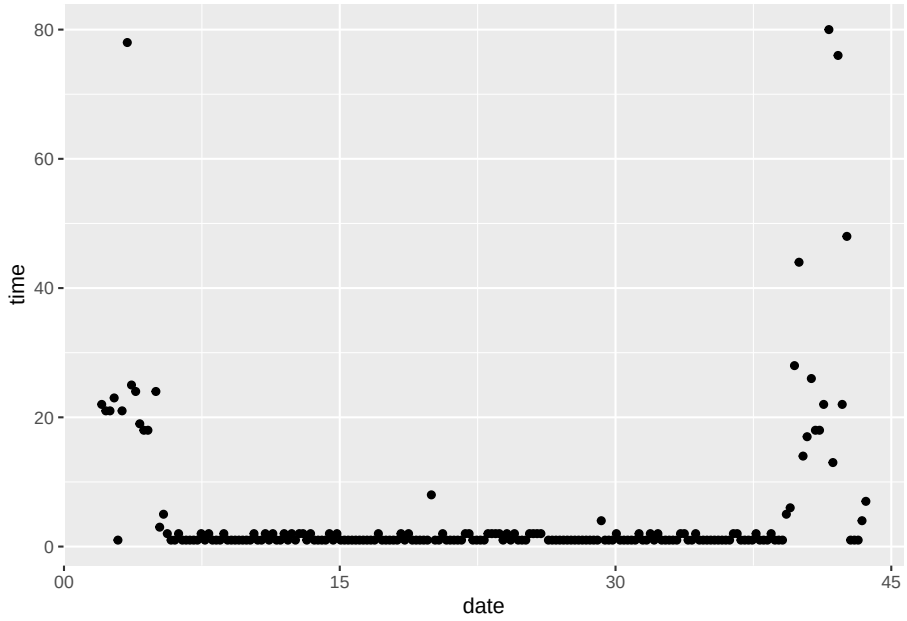
```
library(ggplot2)
ggplot(df,aes(x=date,time))+
  geom_point()
```

```
## Warning: Removed 377 rows containing missing values (‘geom_point()‘).
```



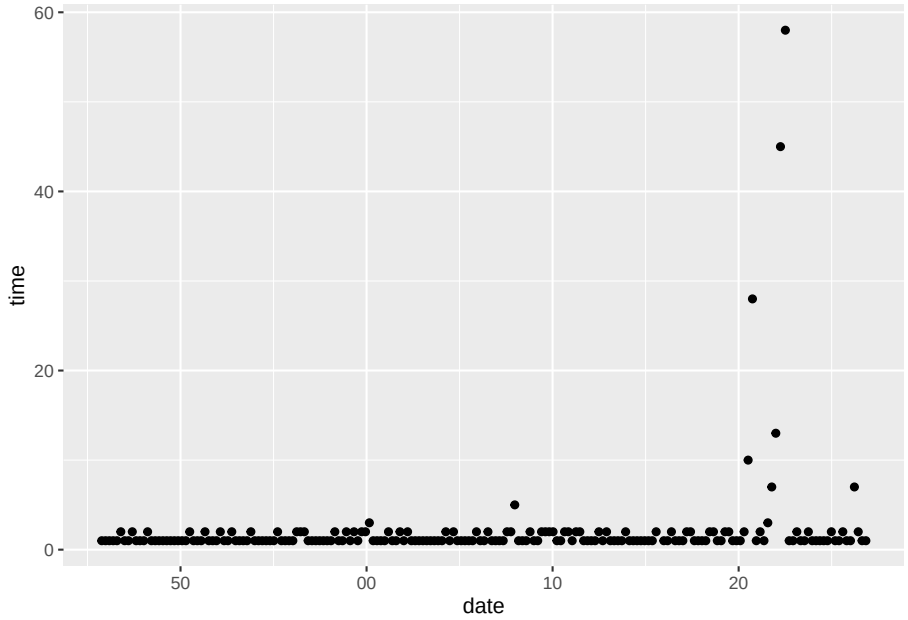

```
ggplot(head(df,200),aes(x=date,time))+
  geom_point()
```

```
## Warning: Removed 2 rows containing missing values (‘geom_point()‘).
```



```
ggplot(tail(df,200),aes(x=date,time))+
  geom_point()
```

```
## Warning: Removed 1 rows containing missing values (‘geom_point()‘).
```

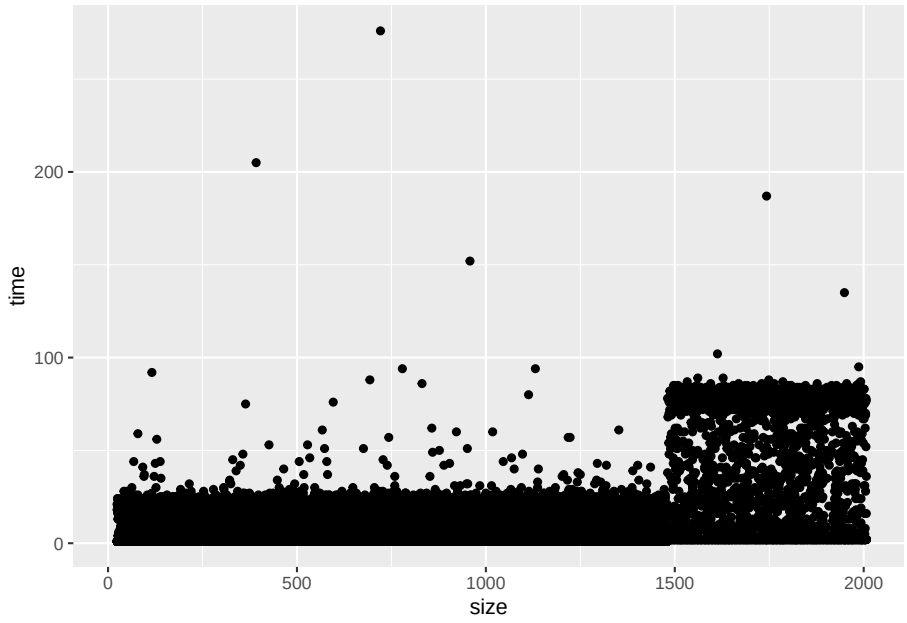


Evolution du temps de transmission en fonction de la taille du paquet

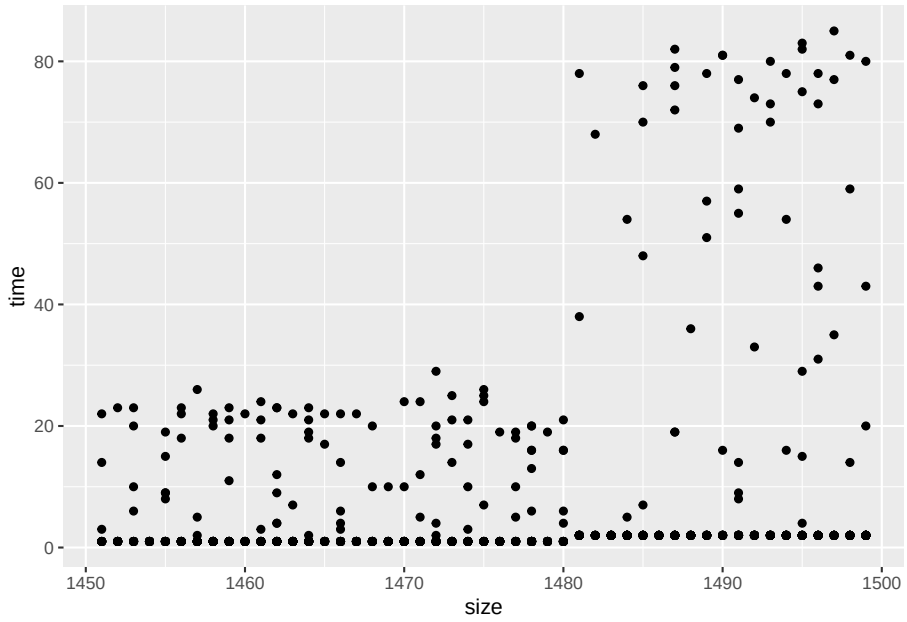
A première vue il semble y avoir une différence de temps à partir de 1480 bytes.

```
library(ggplot2)
ggplot(df,aes(x=size,time))+
  geom_point()
```

```
## Warning: Removed 377 rows containing missing values (‘geom_point()‘).
```




```
ggplot(df[df$size>1450 & df$size<1500,],aes(x=size,time))+
  geom_point()
```



Séparation par taille

On crée donc deux classes.

```
df$class = cut(df$size,breaks = c(0,1480,2020), labels=c("small", "big"))
```

Et on peut représenter désormais l'impact en fonction de la taille et de la classe sur le temps de transmision.

```
library(ggpmisc)
```

```
## Le chargement a nécessité le package : ggpp
```

```
##
## Attachement du package : 'ggpp'
```

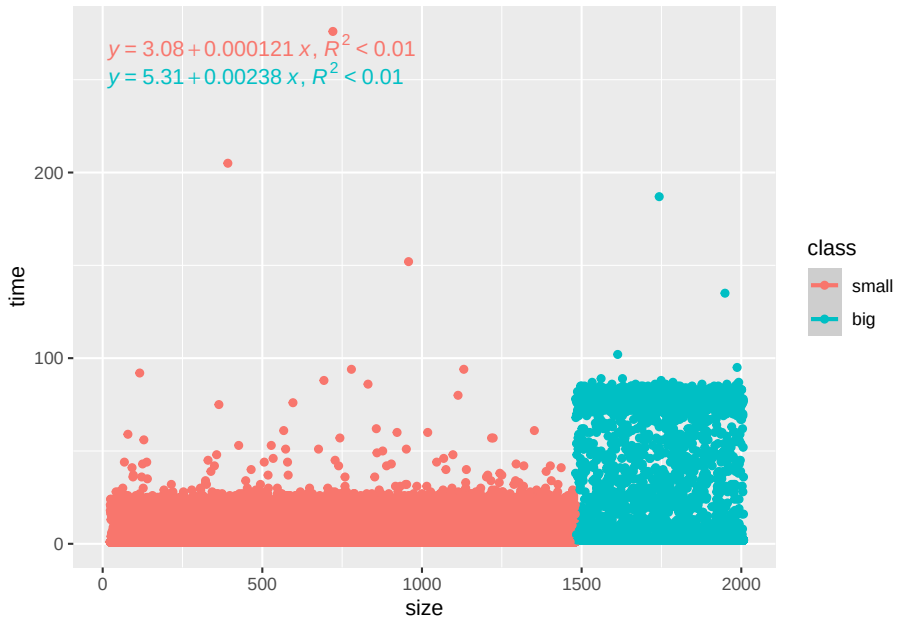
```
## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     annotate
```

```
ggplot(df, aes(x=size, y=time, colour=class))+
  stat_poly_line() +
  stat_poly_eq(aes(label = paste(after_stat(eq.label),
                                 after_stat(rr.label), sep = "*\", \"*"))) +
  geom_point()
```

```
## Warning: Removed 377 rows containing non-finite values (`stat_poly_line()`).
```

```
## Warning: Removed 377 rows containing non-finite values (`stat_poly_eq()`).
```

```
## Warning: Removed 377 rows containing missing values (`geom_point()`).
```



Régression par quantile

La variabilité est tellement forte et asymétrique que la régression du temps moyen peut être considérée comme peu pertinente. On peut vouloir s'intéresser à caractériser plutôt le plus petit temps de transmission. Une approche possible consiste donc à filtrer le plus petit temps de transmission pour chaque taille de message et à effectuer la régression sur ce sous-ensemble de données.

```
library(quantreg)
```

```
## Le chargement a nécessité le package : SparseM

##
## Attachement du package : 'SparseM'

## L'objet suivant est masqué depuis 'package:base':
##
##     backsolve
```

```
modelSmall <- rq(time ~ size , data = df[df$class=="small",], tau = 0.1)
summary(modelSmall, se = "iid")
```

```
##
## Call: rq(formula = time ~ size, tau = 0.1, data = df[df$class == "small",
##     ])
##
## tau: [1] 0.1
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 1.00000 0.41574    2.40535 0.01616
## size        0.00000 0.00049    0.00000 1.00000
```

```
modelBig <- rq(time ~ size , data = df[df$class=="big",], tau = 0.1)
summary(modelBig, se = "iid")
```

```
##
## Call: rq(formula = time ~ size, tau = 0.1, data = df[df$class == "big",
##     ])
##
## tau: [1] 0.1
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept)  2.00000 11.06768    0.18071  0.85660
## size         0.00000  0.00632    0.00000  1.00000
```

Avec cette analyse on remarque que la latence pour le premier quantile est de 1 seconde pour les petits paquets et de 2 secondes pour les gros. La capacité de connexion est aussi différente 0.00049 pour les petits et 0.00632 pour les gros.