Sujet 4 : Estimation de la latence et de la capacité d'une connexion à partir de mesures asymétriques

Samuel MEYNARD

<2020-06-06 Sat>

Contents

1 Énoncé

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.

Votre tâche est d'estimer L et C à partir d'une série d'observations de T pour des valeurs différentes de S. Préparez votre analyse sous forme d'un document computationnel réplicable qui commence avec la lectures des données brutes, disponibles pour deux connexions différentes, qui ont été obtenues avec l'outil ping :

- Le premier jeu de données examine une connexion courte à l'intérieur d'un campus : `http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/liglab2.log.gz`
- Le deuxième jeu de données mesure la performance d'une connexion vers un site Web éloigné assez populaire et donc chargé : `http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/stackoverflow.log.gz`

Les deux fichiers contiennent la sortie brute de l'outil ping qui a été exécuté dans une boucle en variant de façon aléatoire la taille du message. Chaque ligne a la forme suivante:

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

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 !

Votre mission si vous l'acceptez :

1. Commencez par travailler avec le premier jeu de données (liglab2). Représentez graphiquement l'évolution du temps de transmission au cours du temps (éventuellement à différents instants et différentes échelles de temps) pour évaluer la stabilité temporelle du phénomène. Ces variations peuvent-elles être expliquées seulement par la taille des messages ?

2. Représentez le temps de transmission en fonction de la taille des messages. Vous devriez observer une rupture, une taille à partir de laquelle la nature de la variabilité change. Vous estimerez (graphiquement) cette taille afin de traiter les deux classes de tailles de message séparément.

3. Effectuez une régression linéaire pour chacune des deux classes et évaluez les valeurs de L et de C correspondantes. Vous superposerez le résultat de cette régression linéaire au graphe précédent.

4. (Optionnel) 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. Cela peut également être l'occasion pour ceux qui le souhaitent de se familiariser avec la régression de quantiles (implémentée en R dans la bibliothèque quantreg et en Python dans la bibliothèque statsmodels).

5. Répétez les étapes précédentes avec le second jeu de données (stackoverflow)

6. Déposer dans FUN votre résultat

2 Cas 1 : Liglab2

2.1 Récupération du 1er jeu de donnée

2.1.1 Téléchargement

```
from urllib.request import urlretrieve
from os import path
liglab_file = "liglab2.log"
liglab_filegz = liglab_file + ".gz"
url = "http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/liglab2
if not path.exists(liglab_file):
    urlretrieve(url, liglab_filegz)
```

2.1.2 Lecture du fichier

```
import gzip
f = gzip.open(liglab_filegz)
data = f.read().decode('latin-1').strip().splitlines()
f.close()
```

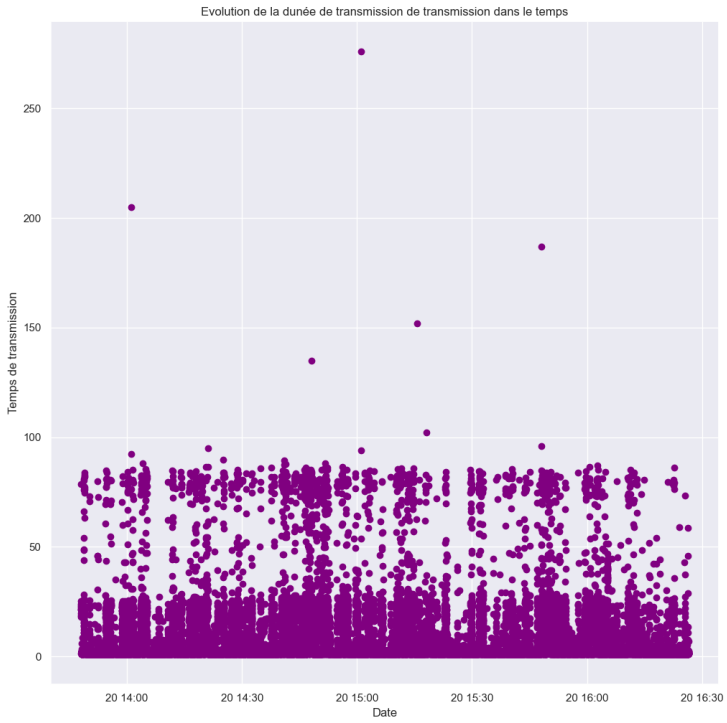
2.1.3 Création d'un tableau sans les données incomplète

```
table = [row.split(' ') for row in data]
cleantable = []
for row in table:
    if len(row) == 10:
        cleantable.append(row)
cleantable[:4]

from datetime import datetime
date = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in cleantable]
S = [int(row[1]) for row in cleantable]
source = [str(row[4]) for row in cleantable]
ip =  [str(row[5][1:-1]) for row in cleantable]
T =  [float(row[8].split('=')[1]) for row in cleantable]
dataset = list(zip(date,donnee, ltime))
T[:10]
```

2.2 Analyse

2.2.1 Cas 1 : evolution de la durée de transmission dans le temps

```
import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(date,
       T,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Date",
       ylabel="Temps de transmission",
       title="Evolution de la dunée de transmission de transmission dans le temps")

plt.savefig('evol_temps_transmission_dans_le_temps.png')
```

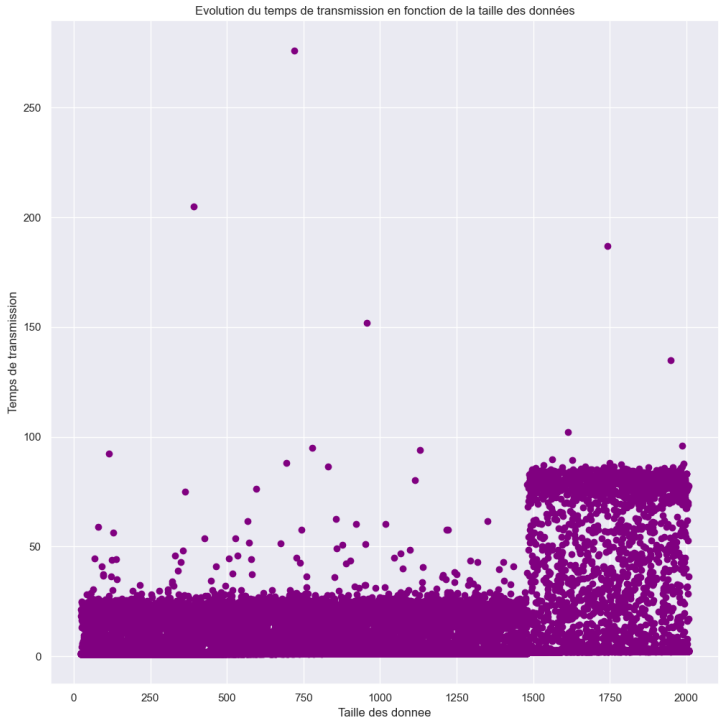


Après analyse visuelle, il ne semble pas avoir d'impact au travers le temps

2.2.2 Cas 2 : Evolution du temps de transmission en fonction de la taille

```
import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(S,
       T,
       color='purple')
```

```
# Set title and labels for axes
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des données")

plt.savefig('evol_temps_transmission_en_fonction_de_la_taille.png')
```



Ici, on voit l'impact de la MTU ici certainement à 1500 sur le temps de transport. Après mesure, celle ci est à 1485.

2.3 Differenciation par rapport à la taille

Nous allons découpé le corpus, en deux sous ensemble. Un inférieur à la MTU et un supérieur.

2.3.1 Inférieur à la MTU

```
table_l1500 = [row for row in cleantable if int(row[1]) <= 1485]
date_l1500 = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in table_l1500]
S_l1500 = [int(row[1]) for row in table_l1500]
T_l1500 =  [float(row[8].split('=')[1]) for row in table_l1500]
dataset_l1500 = list(zip(date_l1500,S_l1500, T_l1500))
dataset_l1500[:10]

import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(S_l1500,
       T_l1500,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des données")

plt.savefig('l1500_evol_T-f(S).png')
```

2.3.2 Supérieur à la MTU

Calcul d'un tableau avec les donnée supérieure à la MTU

```
table_g1500 = [row for row in cleantable if int(row[1]) >= 1485]
date_g1500 = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in table_g1500]
S_g1500 = [int(row[1]) for row in table_g1500]
T_g1500 =  [float(row[8].split('=')[1]) for row in table_g1500]
dataset_g1500 = list(zip(date_g1500,S_g1500, T_g1500))
dataset_g1500[:10]

import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
```

```
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(S_g1500,
       T_g1500,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des données")

plt.savefig('g1500_evol_T-f(S).png')
```

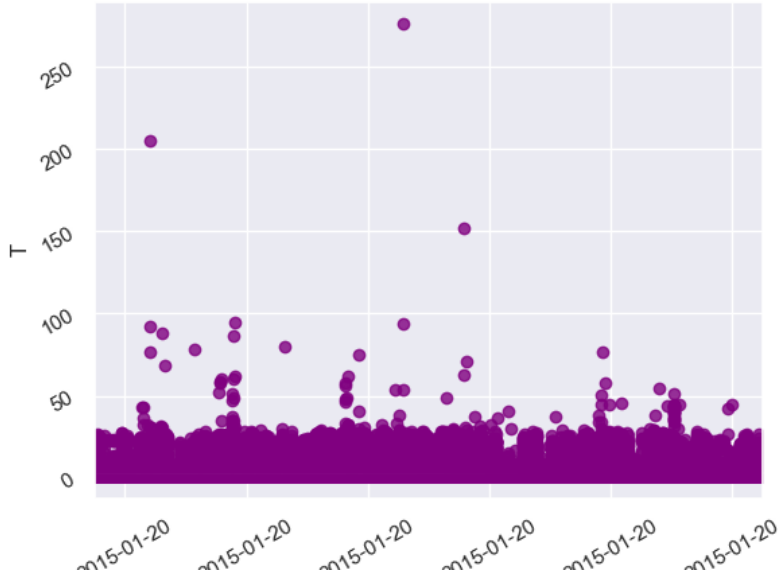
2.4 Régression linéaire

Ici, nous allons procédé à une évaluation de C et L par la technique de la régression linéaire.

2.5 Cas inférieur à la MTU

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import dates
import numpy as np
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from sklearn import linear_model

@plt.FuncFormatter
def fake_dates(x, pos):
    """ Custom formater to turn floats into e.g., 2016-05-08"""
    return dates.num2date(x).strftime('%Y-%m-%d')
sns.set(color_codes=True)
df = pd.DataFrame({
'date': pd.to_datetime(date_l1500),
'datenum': dates.date2num(date_l1500),
'T': T_l1500,
```

```
'S': S_l1500})
fig, ax = plt.subplots()
sns.regplot(x="datenum", y="T", color='purple', data=df, ax=ax)
# here's the magic:
ax.xaxis.set_major_formatter(fake_dates)
# legible labels
ax.tick_params(labelrotation=30)
fig.savefig('l1500_reglineaireT-f(S).png')
```




```
None

np.array(S_l1500).reshape(1, -1)[:9]

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
S_tt = [[value] for value in S_l1500]
my_s = np.array(S_tt)
my_t = np.array(T_l1500)
#my_s = np.array([[1], [2], [3]])
lmodel = LinearRegression()
lmodel.fit(my_s, my_t)
```

```
f"Les coeff sont L = {lmodel.intercept_} et C =  { 1 / lmodel.coef_}"
```

```
Les coeff sont L = 3.257592785874401 et C =  [2761.3155395]
```

2.6 Cas supérieur à la MTU

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
S_tt = [[value] for value in S_g1500]
my_s = np.array(S_tt)
my_t = np.array(T_g1500)
#my_s = np.array([[1], [2], [3]])
lmodel = LinearRegression()
lmodel.fit(my_s, my_t)
f"Les coeff sont L = {lmodel.intercept_} et C =  { 1 / lmodel.coef_}"
```

```
Les coeff sont L = 5.867233082184833 et C =  [441.71908009]
```

3 Cas 2 : Stackoverflow

3.1 Récupération du jeu de donnée

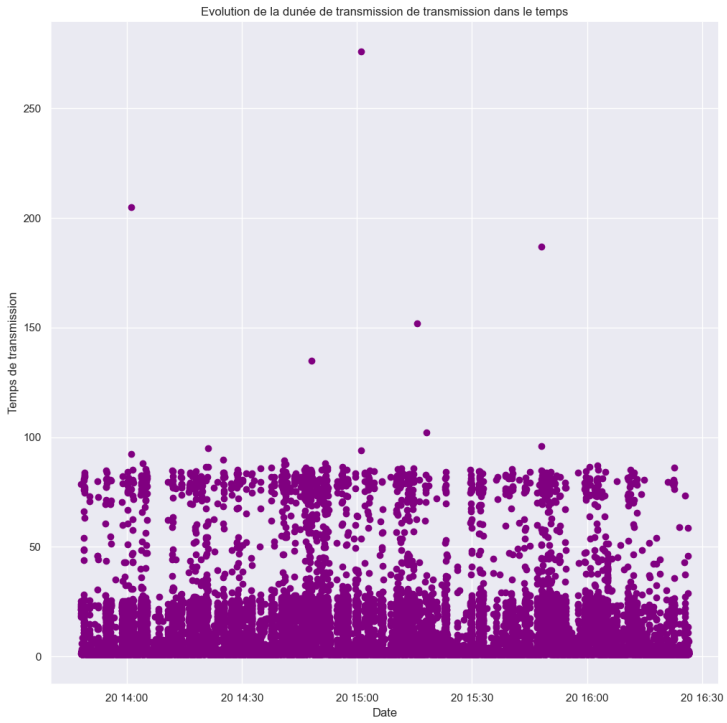
3.1.1 Téléchargement

```
from urllib.request import urlretrieve
from os import path
stacko_file = "stackoverflow.log"
stacko_filegz = stacko_file + ".gz"
url = "http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/stacko
if not path.exists(stacko_file):
    urlretrieve(url, stacko_filegz)
```

3.1.2 Lecture du fichier

```
import gzip
f = gzip.open(stacko_filegz)
data = f.read().decode('latin-1').strip().splitlines()
f.close()
```

```
table = [row.split(' ') for row in data]
cleantable = []
for row in table:
    if len(row) == 10:
        cleantable.append(row)
cleantable[:4]

from datetime import datetime
date = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in cleantable]
S = [int(row[1]) for row in cleantable]
source = [str(row[4]) for row in cleantable]
ip =  [str(row[5][1:-1]) for row in cleantable]
T =  [float(row[8].split('=')[1]) for row in cleantable]
dataset = list(zip(date,donnee, ltime))
T[:10]

import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(date,
       T,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Date",
       ylabel="Temps de transmission",
       title="Evolution de la temps de transmission dans le temps")

plt.savefig('stacko_evol_temps_transmission_dans_le_temps.png')
```

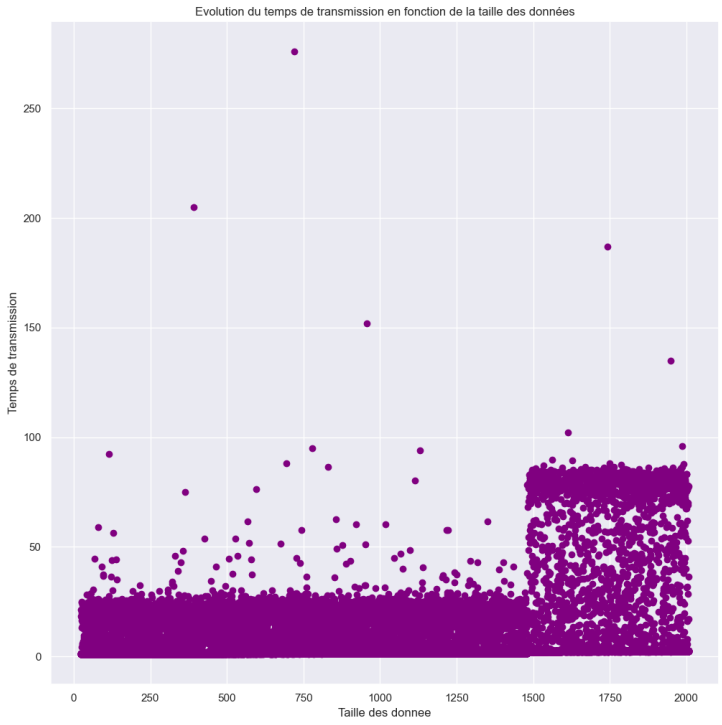


Après analyse visuelle, Il ne semble pas avoir d'impact au travers le temps

3.2 Evolution du temps de transmission à travers le temps

```
import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(S,
       T,
       color='purple')

# Set title and labels for axes
```

```
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des données")

plt.savefig('stacko_evol_temps_transmission_en_fonction_de_la_taille.png')
```



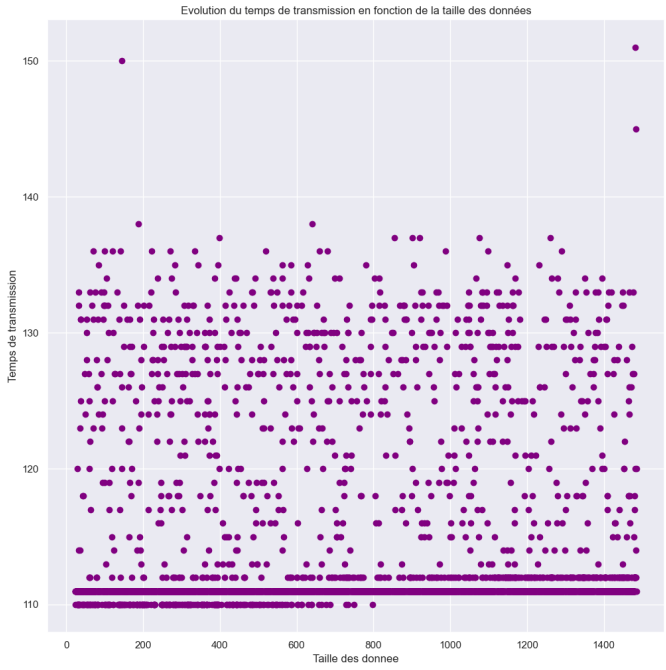
Ici, on voit l'impact de la MTU ici certainement à 1500 sur le temps de transport

3.2.1 Differenciation par rapport à la taille

1. Inférieur à la MTU

```
table_l1500 = [row for row in cleantable if int(row[1]) <= 1485]
date_l1500 = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in table_l150
```

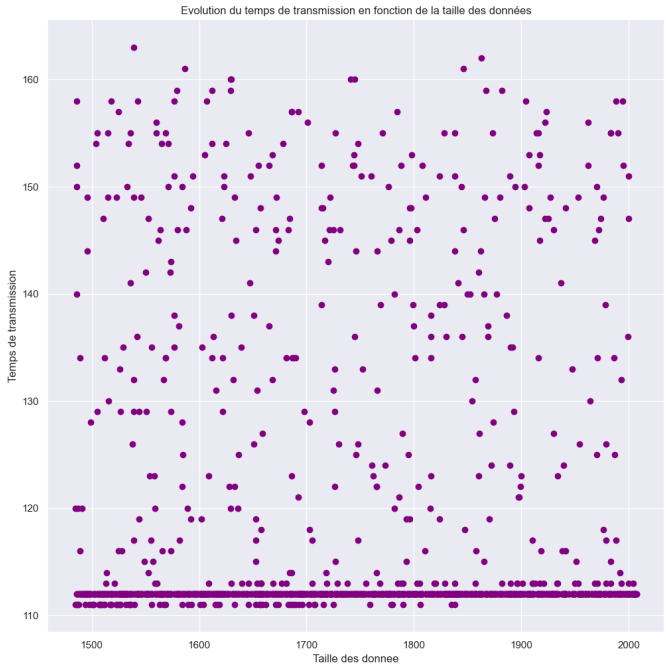
```
S_l1500 = [int(row[1]) for row in table_l1500]
T_l1500 =  [float(row[8].split('=')[1]) for row in table_l1500]
dataset_l1500 = list(zip(date_l1500,S_l1500, T_l1500))
dataset_l1500[:10]

import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
ax.scatter(S_l1500,
       T_l1500,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des donn

plt.savefig('stacko_l1500_evol_T-f(S).png')
```



2. Supérieur à la MTU Calcul d'un tableau avec les donnée supérieure à la MTU

```
table_g1500 = [row for row in cleantable if int(row[1]) >= 1485]
date_g1500 = [datetime.utcfromtimestamp(float(row[0][1:-1])) for row in table_g150
S_g1500 = [int(row[1]) for row in table_g1500]
T_g1500 =  [float(row[8].split('=')[1]) for row in table_g1500]
dataset_g1500 = list(zip(date_g1500,S_g1500, T_g1500))
dataset_g1500[:10]

import matplotlib
import matplotlib.pyplot as plt
# Create figure and plot space
fig, ax = plt.subplots(figsize=(12, 12))

# Add x-axis and y-axis
```

```
ax.scatter(S_g1500,
       T_g1500,
       color='purple')

# Set title and labels for axes
ax.set(xlabel="Taille des donnee",
       ylabel="Temps de transmission",
       title="Evolution du temps de transmission en fonction de la taille des donn

plt.savefig('stacko_g1500_evol_T-f(S).png')
```

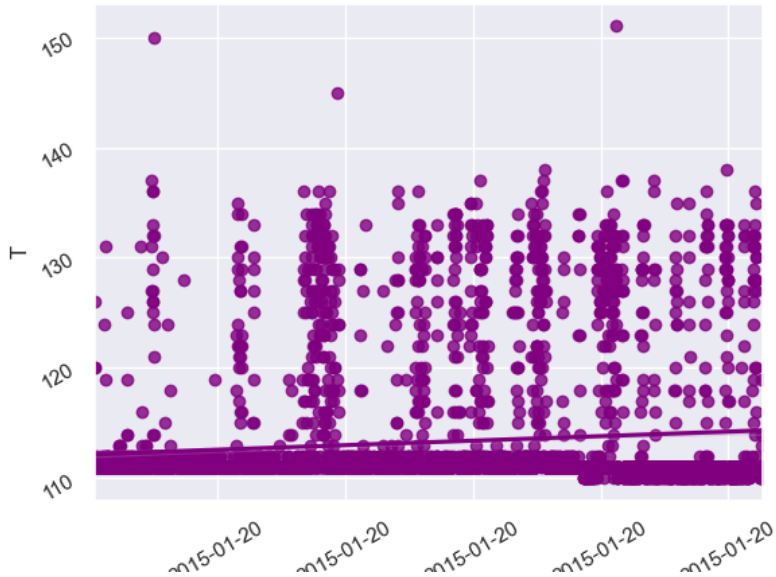


3.3 Régression linéaire

3.3.1 Cas inférieur à la MTU

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
from matplotlib import dates
import numpy as np
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from sklearn import linear_model

@plt.FuncFormatter
def fake_dates(x, pos):
    """ Custom formater to turn floats into e.g., 2016-05-08"""
    return dates.num2date(x).strftime('%Y-%m-%d')
sns.set(color_codes=True)
df = pd.DataFrame({
'date': pd.to_datetime(date_l1500),
'datenum': dates.date2num(date_l1500),
'T': T_l1500,
'S': S_l1500})
fig, ax = plt.subplots()
sns.regplot(x="datenum", y="T", color='purple', data=df, ax=ax)
# here's the magic:
ax.xaxis.set_major_formatter(fake_dates)
# legible labels
ax.tick_params(labelrotation=30)
fig.savefig('stacko_l1500_reglineaireT-f(S).png')
```




```
None
```

```
np.array(S_l1500).reshape(1, -1)[:9]
```

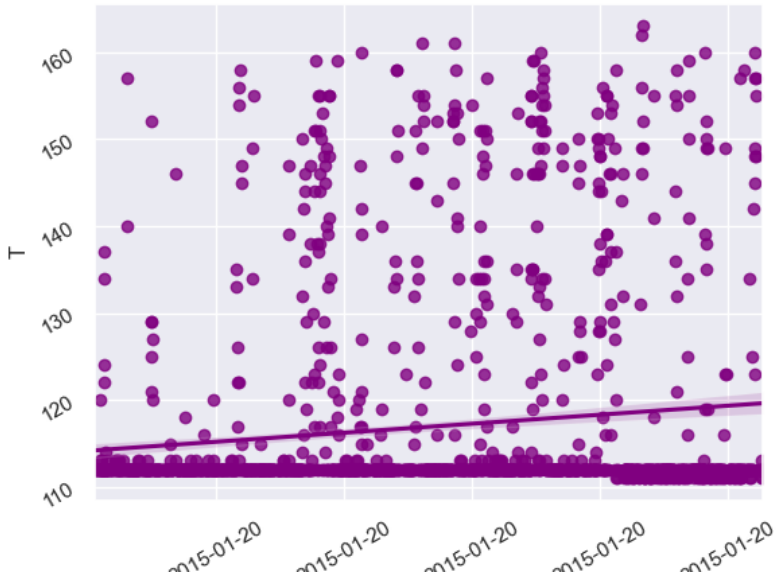
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
S_tt = [[value] for value in S_l1500]
my_s = np.array(S_tt)
my_t = np.array(T_l1500)
#my_s = np.array([[1], [2], [3]])
lmodel = LinearRegression()
lmodel.fit(my_s, my_t)
f"Les coeff sont L = {lmodel.intercept_} et C =  { 1 / lmodel.coef_}"
```

```
Les coeff sont L = 113.19744441078485 et C =  [9479.50730539]
```

3.3.2 Cas supérieur à la MTU

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import dates
```

```
import numpy as np
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from sklearn import linear_model

@plt.FuncFormatter
def fake_dates(x, pos):
    """ Custom formater to turn floats into e.g., 2016-05-08"""
    return dates.num2date(x).strftime('%Y-%m-%d')
sns.set(color_codes=True)
df = pd.DataFrame({
'date': pd.to_datetime(date_g1500),
'datenum': dates.date2num(date_g1500),
'T': T_g1500,
'S': S_g1500})
fig, ax = plt.subplots()
sns.regplot(x="datenum", y="T", color='purple', data=df, ax=ax)
# here's the magic:
ax.xaxis.set_major_formatter(fake_dates)
# legible labels
ax.tick_params(labelrotation=30)
fig.savefig('stacko_g1500_reglineaireT-f(S).png')
```




```
None
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
S_tt = [[value] for value in S_g1500]
my_s = np.array(S_tt)
my_t = np.array(T_g1500)
#my_s = np.array([[1], [2], [3]])
lmodel = LinearRegression()
lmodel.fit(my_s, my_t)
f"Les coeff sont L = {lmodel.intercept_} et C =  { 1 / lmodel.coef_}"
```

Etonnant pour C négatif alors que l'analyse graphique montre une pente positivew