

CELL

Mobile data analytics

Thi-Mai-Trang Nguyen
LIP6-Sorbonne Université

Plan

- **Call Detail Record**
- Telecom Italia dataset
- Python
- Matlab
- C-RAN dimensioning

Call Detail Record (CDR)

- Data on mobile traffic (call, SMS, Internet) are usually collected by mobile operator in the form of Call Detail Record (CDR)
- Used for billing and network management
- There exist several types of CDR
- An example of CDR
 - Timestamp
 - Caller_id
 - Callee_id
 - Call_duration
 - Antenna_code

Data structure

- Dataset usually follows the Tabulation Separated Values (TSV) format
 - Columns are separated by tabulation
- Each dataset has a document which explains the meaning of each column
- Example

```
CREATE TABLE H_A_FLOWS (  
  date_hour TIMESTAMP,  
  originating_ant INTEGER,  
  terminating_ant INTEGER,  
  nb_voice_calls INTEGER,  
  duration_voice_calls INTEGER  
);
```

```
2012-04-28 23:00:00 1236 786 2 96  
2012-04-28 23:00:00 1236 804 1 539  
2012-04-28 23:00:00 1236 867 3 1778  
2012-04-28 23:00:00 1236 939 1 1  
2012-04-28 23:00:00 1236 1020 6 108  
2012-04-28 23:00:00 1236 1065 1 1047  
2012-04-28 23:00:00 1236 1191 1 67  
2012-04-28 23:00:00 1236 1236 18 2212  
2012-04-28 23:00:00 1237 323 1 636  
2012-04-28 23:00:00 1237 710 1 252
```

Antenna position

- The position of an antenna in a mobile network is described by its GPS (Global Positioning System) coordinates
- Example

```
CREATE TABLE ANT_POS(  
  antenna_id INTEGER,  
  longitude FLOAT,  
  latitude FLOAT,  
);
```

```
1 -4.143452 5.342044  
2 -3.913602 5.341612  
3 -3.967045 5.263331  
4 -4.070007 5.451365  
5 -3.496235 6.729410  
6 -3.485944 6.729422  
7 -3.981175 5.273144  
8 -3.911705 5.858010  
9 -4.014445 5.421120
```

Plan

- Call Detail Record
- **Dataset - Telecom Italia**
- Python
- Matlab
- C-RAN dimensioning

Telecom Italia Open Big Data











- In 2014, Telecom Italia launches the first edition of Big Data Challenge

<https://dandelion.eu/datamine/open-big-data/>

- Two datasets on call, SMS and Internet connection
 - Milan
 - Trento
- 2 months of data collected in November and December 2013

File

- Each file contains data of one-day traffic
- File size: 270 - 350 MB

 sms-call-internet-mi-2013-12-01	14/02/2017 16:08	Document texte	291 715 Ko
 sms-call-internet-mi-2013-12-02	14/02/2017 16:08	Document texte	333 906 Ko
 sms-call-internet-mi-2013-12-03	14/02/2017 16:08	Document texte	345 652 Ko
 sms-call-internet-mi-2013-12-04	14/02/2017 16:08	Document texte	343 782 Ko
 sms-call-internet-mi-2013-12-05	14/02/2017 16:08	Document texte	345 234 Ko
 sms-call-internet-mi-2013-12-06	14/02/2017 16:08	Document texte	345 731 Ko
 sms-call-internet-mi-2013-12-07	14/02/2017 16:08	Document texte	299 973 Ko
 sms-call-internet-mi-2013-12-08	14/02/2017 16:08	Document texte	286 781 Ko
 sms-call-internet-mi-2013-12-09	14/02/2017 16:08	Document texte	331 019 Ko
 sms-call-internet-mi-2013-12-10	14/02/2017 16:09	Document texte	341 084 Ko

Data in a file

- Data in each file follows the TSV format with 8 columns
 - **Square ID**: Identifier of a cell (a square) in the city map
 - **Timestamp**: the starting time of a 10-minute time interval calculated in ms since 01/01/1970 UTC
 - **Country code**: the country code following the ITU-T E.164 standard
 - **SMS-in**: number of incoming SMSs received in the square, originated from the indicated country
 - **SMS-out** : number of outgoing SMSs sent to the indicated country
 - **Call-in** : number of incoming calls in the square originated from the indicated country
 - **Call-out** : number of outgoing calls in the square towards the indicated country
 - **Internet** : volume of Internet traffic measured in the square, generated by users identified by country code

1	1385853000000	39	0.16513682662061693	0.1763994583739133	0.030875085088185057	0.02730046487718618	13.330858194494864
1	1385853600000	0	0.029087774982685617	0.02730046487718618			
1	1385853600000	39	0.18645109168870494	0.13658782275823106	0.05460092975437236	11.329552259939573	
1	1385854200000	39	0.21965225800268914	0.38112896604000707	0.0825256641241584	0.13596355326563117	13.16616288096205
1	1385854800000	39	0.29511418007714996	0.11045039849394445	0.05460092975437236	0.07957531340575938	13.32165338919027
1	1385854800000	46		0.026137424264286602			
1	1385855400000	39	0.052274848528573205	0.13542478214533146	0.05343788914147278	0.02730046487718618	12.434743835804813
1	1385856000000	0	0.026137424264286602				
1	1385856000000	39	0.13533928377303134	0.12117425912694108	0.07957531340575938	0.0563882398598718	13.860352866846307
1	1385856600000	39	0.05460092975437236	0.05522519924697222	0.02730046487718618	0.029087774982685617	12.557325871001357

Missing data

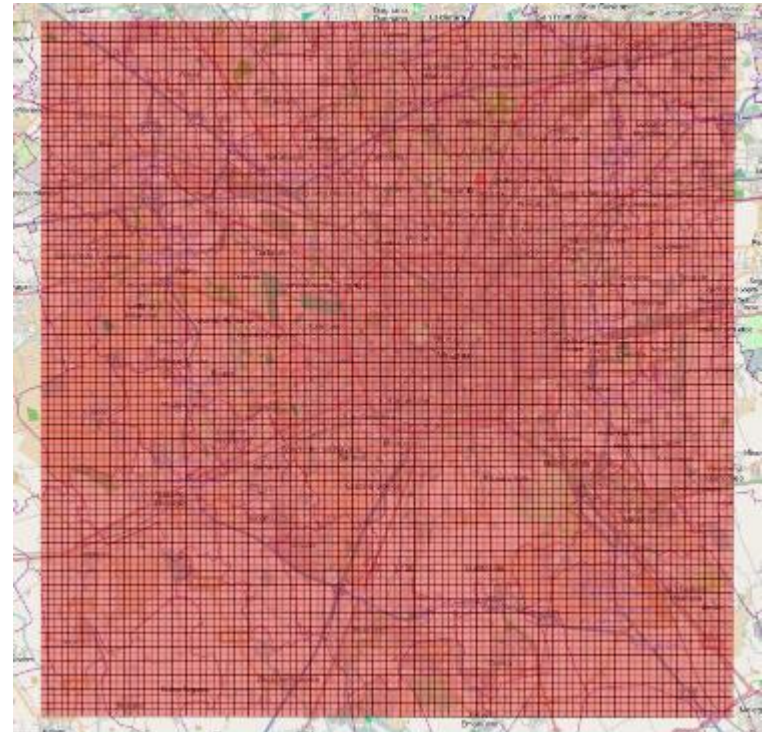
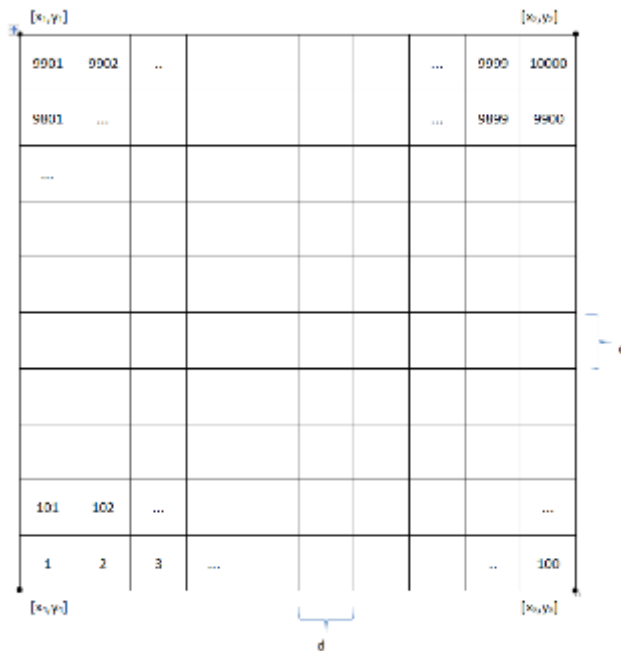
- When there is no activity for call/SMS/Internet in a time interval, the corresponding values are missing
- Example: a line missing SMS-in activity

s \t i \t c \t \t SMSout \t Callin \t Callout \t Internettraffic

- A time interval which doesn't have any activity (no call, no SMS, no Internet connection) is not registered in the data set

Cell ID (Square ID)

- The map of each city (Milan or Trento) is divided into 10000 square cell by a 100 x 100 grid
- Each cell has a cell-Id (Square ID)



Understand the data

- Data registered in each line is calculated as an aggregation of Call Detail Record (CDR) generated by the Telecom Italia network
- SMS-in / SMS-out
 - A CDR is generated every time a user receives an SMS (SMS-in) or sends an SMS (SMS-out)
- Call-in / Call-out
 - A CDR is generated every time a user is called (Call-in) or makes a call (Call-out)
- Internet connection
 - A CDR is generated when a user starts an Internet connection or terminates an Internet connection.
 - During the Internet connection, an additional CDR is generated when its duration reaches 15 mn or 5 MB since its last CDR
- The data is normalized for confidentiality reasons

Plan

- Call Detail Record
- Dataset - Telecom Italia
- **Python**
- Matlab
- C-RAN dimensioning

NumPy

- With NumPy, use function `loadtxt()` to load a data file
- This function suppose that there is not header line and all data in the file has the same format
- Exemple

1	1385854200000	39	0.21965225800268914	0.38112896604000707	0.0825256641241584	0.13596355326563117	13.16616288096205
1	1385854800000	39	0.29511418007714996	0.11045039849394445	0.05460092975437236	0.07957531340575938	13.32165338919027
1	1385855400000	39	0.052274848528573205	0.13542478214533146	0.05343788914147278	0.02730046487718618	12.434743835804813
1	1385856000000	39	0.13533928377303134	0.12117425912694108	0.07957531340575938	0.0563882398598718	13.860352866846307
1	1385856600000	39	0.05460092975437236	0.05522519924697222	0.02730046487718618	0.029087774982685617	12.557325871001357
1	1385857200000	39	0.13301320254723217	0.1633495165151175	0.02730046487718618	0.026137424264286602	12.644994510871808
1	1385858400000	39	0.026137424264286602	0.02792473436978604	0.02730046487718618	0.1365023243859309	11.266504250571113
1	1385859000000	39	0.05343788914147278	0.11992572014174135	0.02730046487718618	0.05460092975437236	10.106504550041493
1	1385863800000	39	0.10920185950874473	0.2440023721614763	0.026137424264286602	0.05343788914147278	6.95683973836449
1	1385865600000	39	0.4095069731577927	0.0017873101054994376	0.07841227279285981	0.02730046487718618	6.879137233436531

```
import numpy as np
d = np.loadtxt('data2.txt', delimiter='\t')
print(d.shape)
```

```
(10, 8)
```

Basic data analytics (1)

- NumPy has functions for basic data analytics
- Change the display format so that the results are easy to read

```
np.set_printoptions(formatter={'float': '{: 0.3f}'.format})
```

- To calculate the sum of each column

```
print(d.sum(axis=0))
```

```
[ 10.000 13858581000000.000 390.000  1.488  1.360  0.484  0.626 113.194]
```

Basic data analytics (2)

- To calculate the average value of each feature (each column)

```
print(d.mean(axis=0))
```

```
[ 1.000 1385858100000.000 39.000 0.149 0.136 0.048 0.063 11.319]
```

- To calculate the standard deviation of each column

```
print(d.std(axis=0))
```

```
[ 0.000 3612478.374 0.000 0.118 0.105 0.023 0.040 2.421]
```

- To display the maximum and minimum values of each column

```
print(d.max(axis=0))
```

```
print(d.min(axis=0))
```

```
[ 1.000 1385865600000.000 39.000 0.410 0.381 0.083 0.137 13.860]
[ 1.000 1385854200000.000 39.000 0.026 0.002 0.026 0.026 6.879]
```


Pandas

- With Pandas, use function `read_csv()` to load the dataset
- This function does not require that the data must have the same type in a file
- Example

```
from pandas import read_csv
f = "data2.txt"
n = ['sqid', 'time', 'country', 'sms-in', 'sms-out', 'call-in', 'call-out', 'internet']
d = read_csv(f, names=n, sep='\t')
print(d.shape)
```

(10, 8)

Descriptive statistics

- Function `describe()` lists the statistical properties of each attribute
- Example

```
print(d.describe())
```

	sqid	time	country	sms-in	sms-out	call-in	call-out	internet
count	10.0	1.000e+01	10.0	10.000	10.000	10.000	10.000	10.000
mean	1.0	1.386e+12	39.0	0.149	0.136	0.048	0.063	11.319
std	0.0	3.808e+06	0.0	0.124	0.110	0.024	0.043	2.552
min	1.0	1.386e+12	39.0	0.026	0.002	0.026	0.026	6.879
25%	1.0	1.386e+12	39.0	0.054	0.069	0.027	0.028	10.397
50%	1.0	1.386e+12	39.0	0.121	0.121	0.040	0.054	12.496
75%	1.0	1.386e+12	39.0	0.199	0.156	0.072	0.074	13.036
max	1.0	1.386e+12	39.0	0.410	0.381	0.083	0.137	13.860

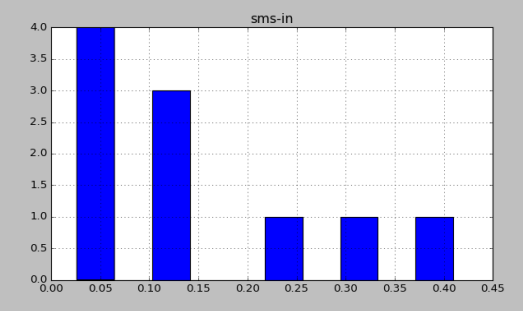
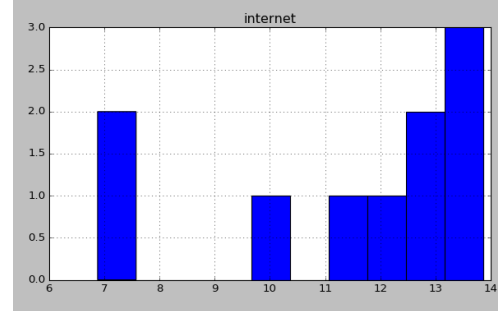
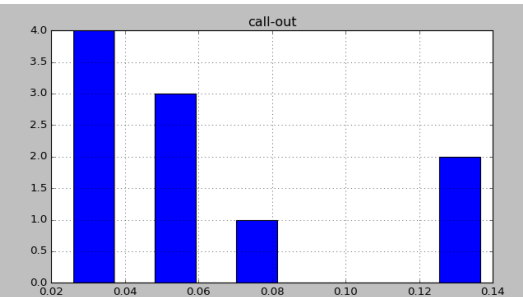
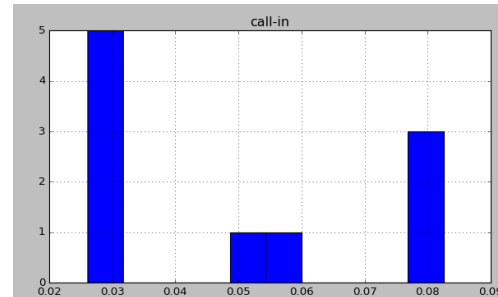
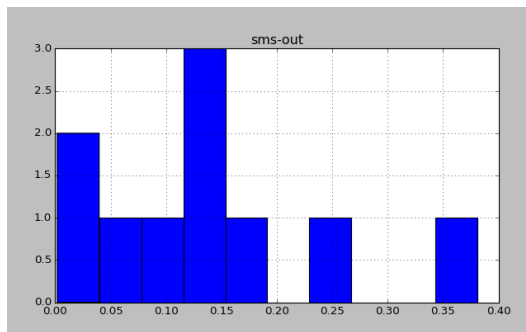
Data visualization

- `Matplotlib` allows us to observe the distribution of the data in different forms
 - Histogram
 - Density
 - Box and Whisker

Histogram

- Histogram groups data into bins and provides you the number of observations in each bin

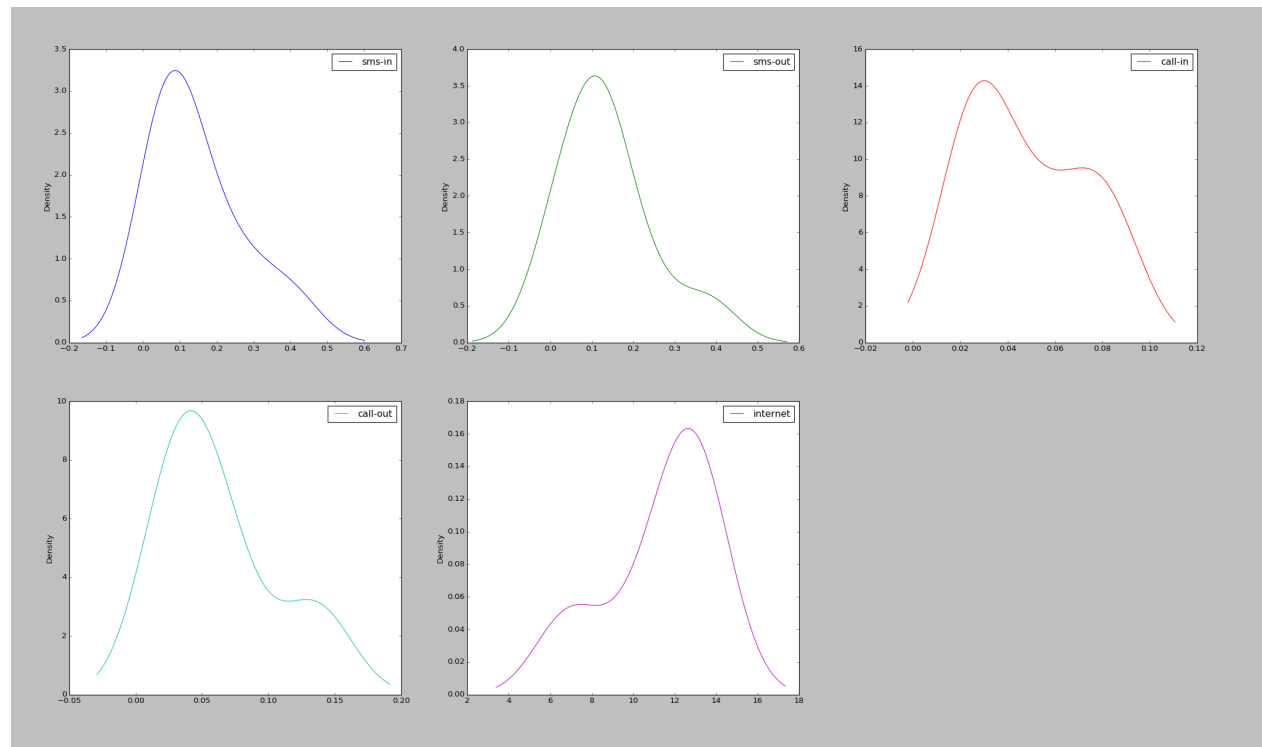
```
from matplotlib import pyplot  
d.hist()  
pyplot.show()
```



Density plot

- Density plots show the distribution of each attribute using a smooth curve through the top of each bin

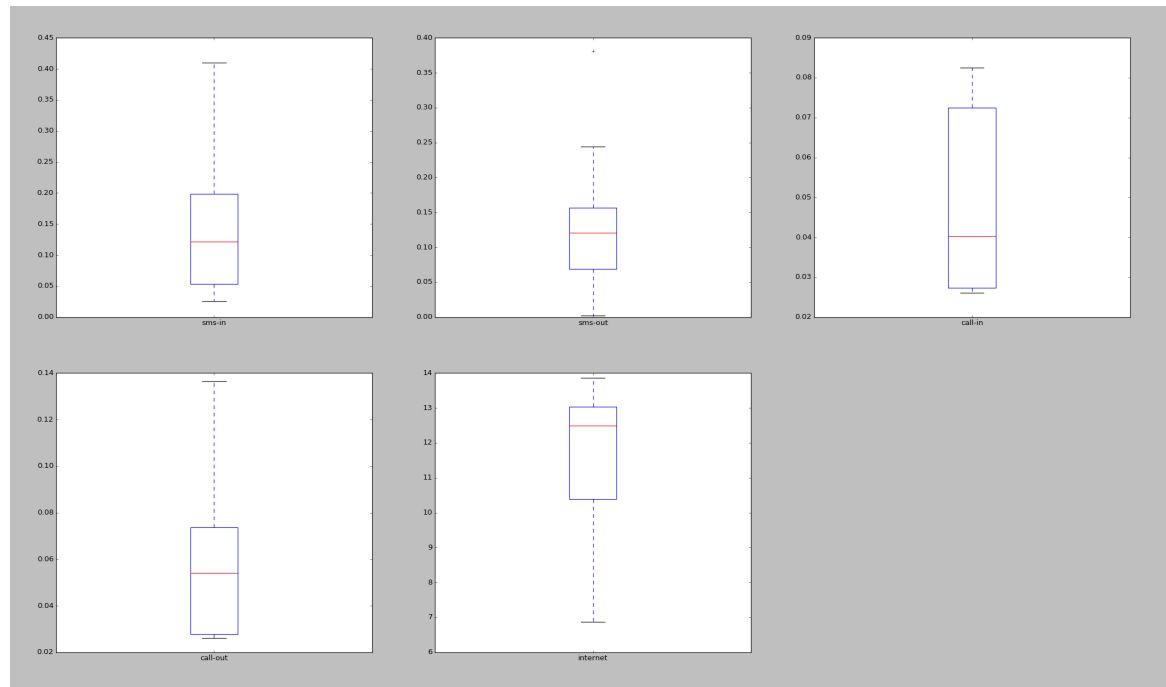
```
d.plot(kind='density', subplots=True, layout=(2,3), sharex=False)  
pyplot.show()
```



Box-and-Whisker plot

- The Box shows the median, the 25th and the 75th percentiles
- The Whiskers show the minimum and maximum values (or outliers) of the observed data

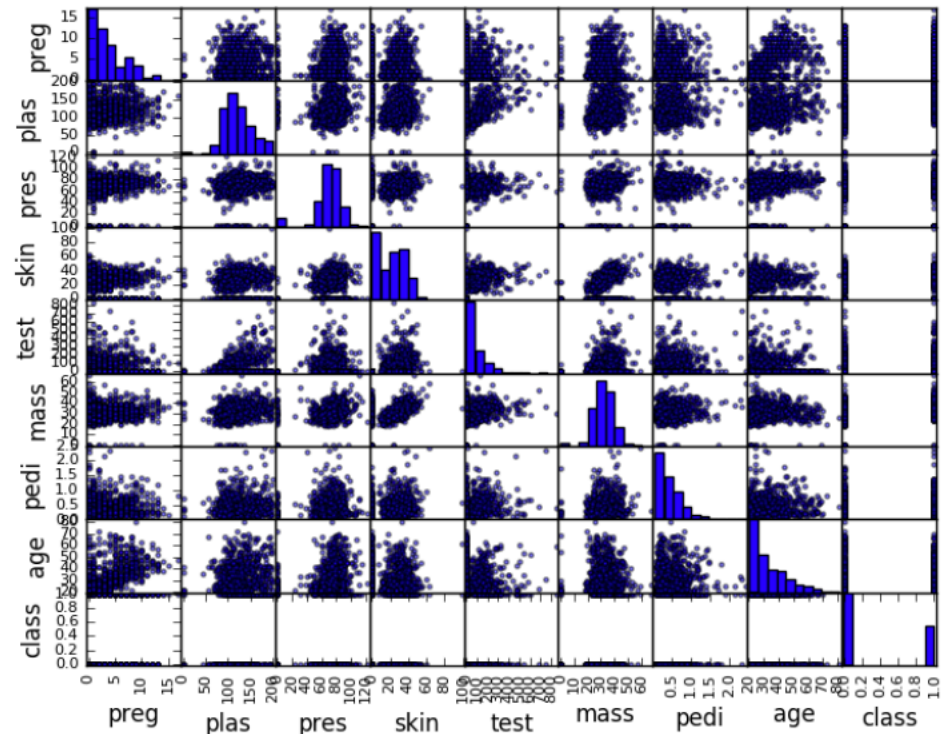
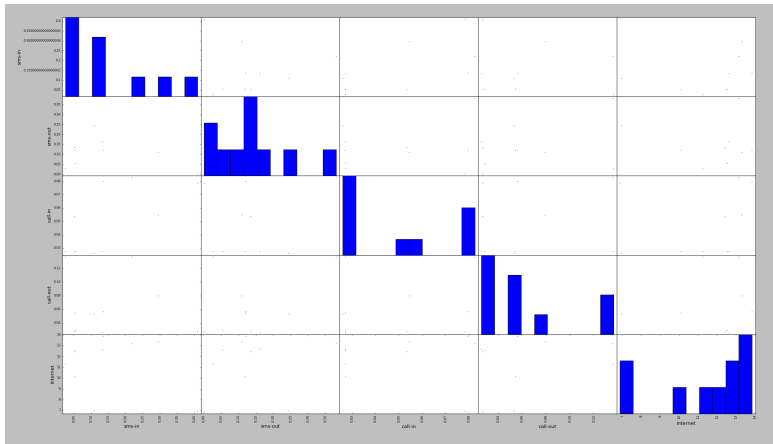
```
d.plot(kind='box', subplots=True, layout=(2,3), sharex=False, sharey=False)  
pyplot.show()
```



Scatter plots

- Scatter plots present the relationship between two variables as dots in two dimensions, one axis for each attribute

```
from pandas.plotting import scatter_matrix
scatter_matrix(d)
pyplot.show()
```



Plan

- Call Detail Record
- Jeu de données - Telecom Italia
- Python
- **Matlab**
- C-RAN dimensioning

Load a dataset

- In Matlab, function `load` loads a dataset in the form of columns and creates variables
- Example

0.21965225800268914	0.38112896604000707	0.0825256641241584	0.13596355326563117	13.16616288096205
0.29511418007714996	0.11045039849394445	0.05460092975437236	0.07957531340575938	13.32165338919027
0.052274848528573205	0.13542478214533146	0.05343788914147278	0.02730046487718618	12.434743835804813
0.13533928377303134	0.12117425912694108	0.07957531340575938	0.0563882398598718	13.860352866846307
0.05460092975437236	0.05522519924697222	0.02730046487718618	0.029087774982685617	12.557325871001357
0.13301320254723217	0.1633495165151175	0.02730046487718618	0.026137424264286602	12.644994510871808
0.026137424264286602	0.02792473436978604	0.02730046487718618	0.1365023243859309	11.266504250571113
0.05343788914147278	0.11992572014174135	0.02730046487718618	0.05460092975437236	10.106504550041493
0.10920185950874473	0.2440023721614763	0.026137424264286602	0.05343788914147278	6.95683973836449
0.4095069731577927	0.0017873101054994376	0.07841227279285981	0.02730046487718618	6.879137233436531

`load data3.txt`

Data statistics

- Maximum value

```
mx = max(data3) →
```

0.4095	0.3811	0.0825	0.1365	13.8604
--------	--------	--------	--------	---------

- Average value

```
mu = mean(data3) →
```

0.1488	0.1360	0.0484	0.0626	11.3194
--------	--------	--------	--------	---------

- Median value

```
me = median(data3) →
```

0.1211	0.1205	0.0404	0.0540	12.4960
--------	--------	--------	--------	---------

- Minimum value

```
mi = min(data3) →
```

0.0261	0.0018	0.0261	0.0261	6.8791
--------	--------	--------	--------	--------

- Standard deviation

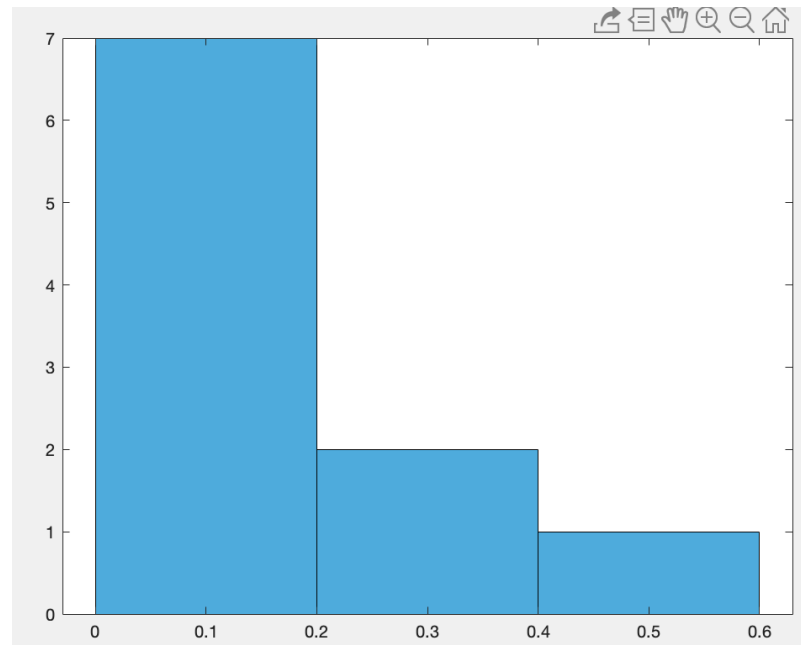
```
sigma = std(data3) →
```

0.1242	0.1105	0.0244	0.0425	2.5524
--------	--------	--------	--------	--------

Histogram

- We can extract a column (a feature) to observe its histogram
- Example : histogram of SMS-in

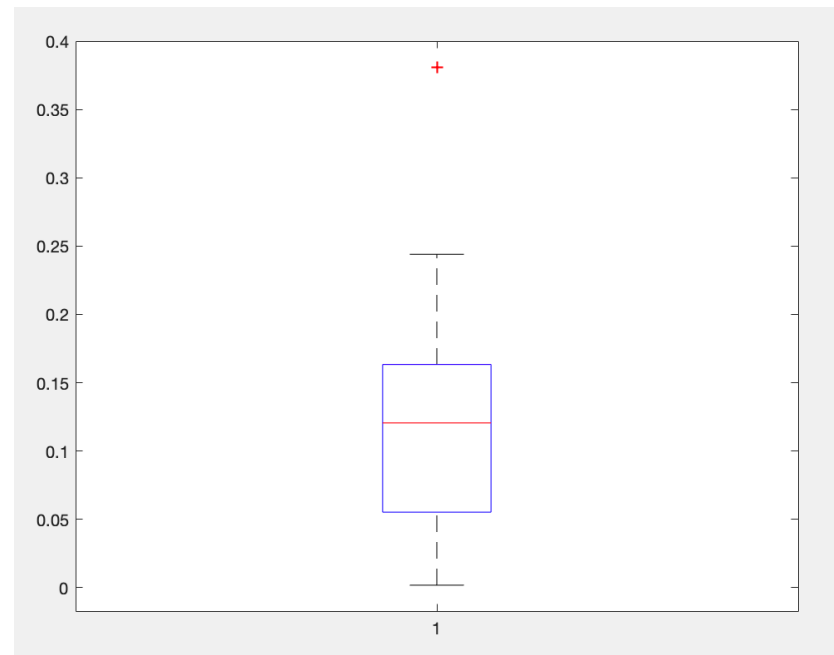
```
[n,p] = size(data3)
t = 1:p;
si = data3(:,1);
histogram(si)
```



Boxplot

- Function `boxplot()` shows the statistics of a feature in the form of Box and Whisker
- Example : Box and Whisker plot of SMS-out

```
so = data3(:,2);  
boxplot(so)
```

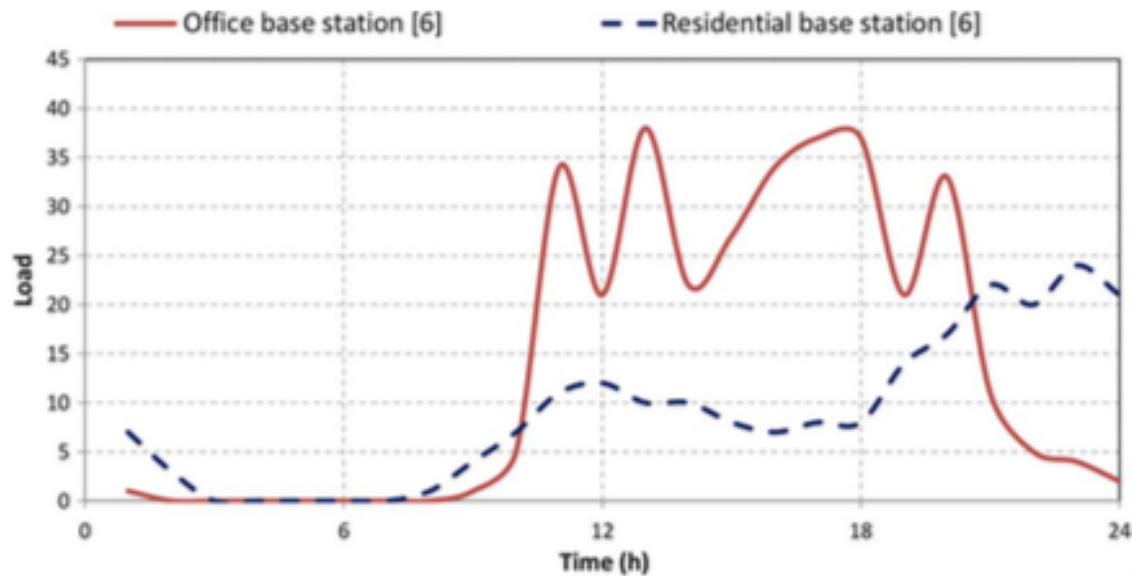


Plan

- Call Detail Record
- Jeu de données - Telecom Italia
- Python
- Matlab
- **C-RAN dimensioning**

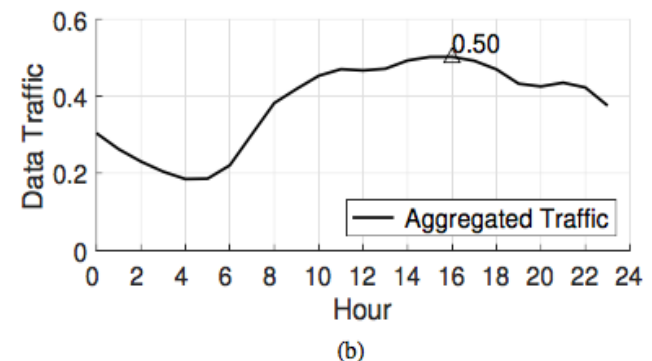
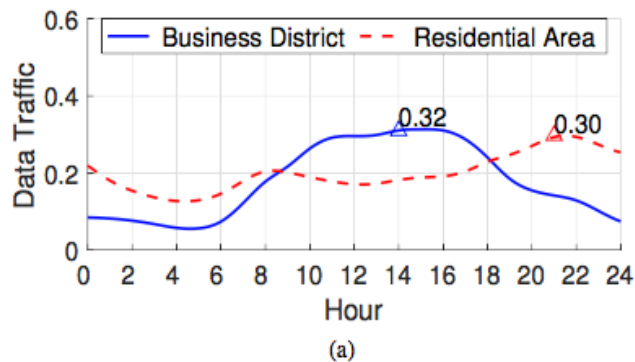
Mobile traffic demand

- Mobile traffic is not uniform
 - Spatio: office vs. residential areas
 - Temporal: peak hours vs. off-peak hours



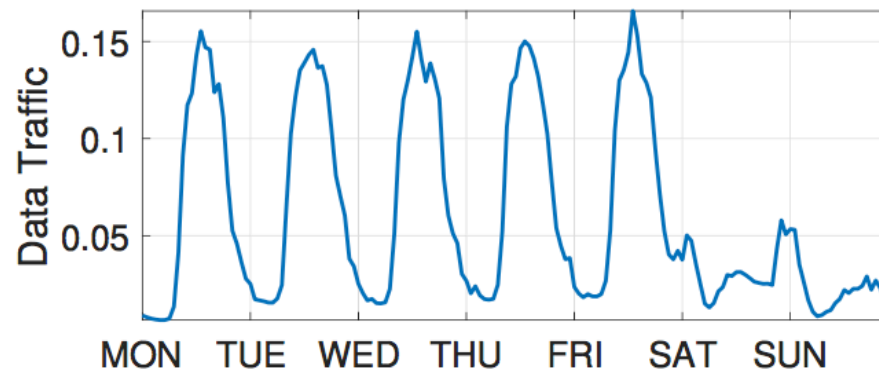
Base station clustering

- Base stations having complementary traffic profiles are assigned to the same BBU Pool
 - Aggregated traffic is more stable → increased resource utilization rate → reduced energy consumption
 - Non-overlapping peaks → reused peak capacity → reduced deployment cost



Traffic profile of a base station

- Traffic profile depends on
 - Weekday / weekend
 - Morning / Afternoon / Evening
 - Residential / business areas



How to represent traffic profile?

- For each base station
 - A vector having 24 elements is used to represent the traffic profile of a weekday
 - Each element corresponds to traffic generated during one hour of a day

$$\mathbf{f}_w(s_i) = [u_1, u_2, \dots, u_{24}]$$

- Another vector of 24 elements is used to represent the traffic profile of a weekend day

$$\mathbf{f}_n(s_i) = [v_1, v_2, \dots, v_{24}]$$

- A vector of 48 elements which is the concatenation of the two previous vectors represents the traffic profile of each base station

$$\mathbf{f}(s_i) = [\mathbf{f}_w(s_i), \mathbf{f}_n(s_i)]$$

Base station complementarity (1)

- Base stations having non-overlapping peak-times in their traffic profiles are complementary
 - Extract peak-times of each base station into a vector

$$T(s_i) = \{t_{i_1}, t_{i_2}, \dots, t_{i_m}\}, \quad 1 \leq i_m \leq 24$$

- Calculate the Shannon entropy of each base station group (≥ 2 base stations)

$$H(S) = - \sum_{k=1}^K p_k \log p_k$$

→ The higher $H(S)$ is → the more base stations are complementary

Base station complementarity (2)

- The more aggregated traffic is stable → the better network capacity utilization is

- To measure the capacity utilization rate of the BBU Pool

$$U(S) = \frac{\text{mean } \mathbf{f}(S)}{\max \mathbf{f}(S)}$$

- where

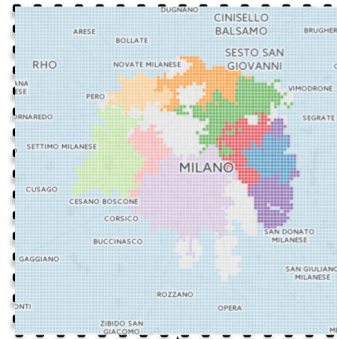
- $\mathbf{f}(S)$ is the profile of the aggregated traffic of base station cluster

- To take into account both complementarity and stability → define a new metric

$$M(S) = U(S) * H(S) = -\frac{\text{mean } \mathbf{f}(S)}{\max \mathbf{f}(S)} \sum_{k=1}^K p_k \log p_k$$

C-RAN dimensioning problem

- Given a city with traffic profiles and base station coordinates
 - What is the optimal number of BBU Pools?
 - What is the optimal RRH-BBU Pool mapping?
- This is a clustering problem
 - Network is modeled as a graph $G = (V, E)$
 - Each base station is a node
 - If the distance between two base stations is smaller than a threshold
→ there is an edge between these two nodes
 - Link cost = $M(S)$ calculated over these two base stations
 - Find the optimal number of clusters and the optimal base station assignment to maximize the capacity utilization rate and to minimize the required capacity of the BBU Pools

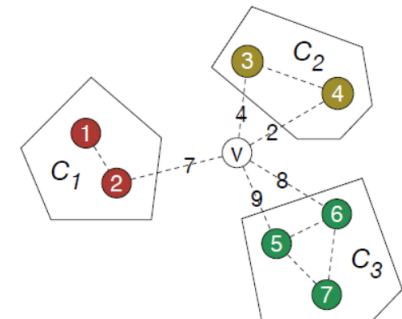


Clustering algorithm

- Initialization
 - Each base station is a labeled cluster in a list (L)
- Iteration
 - Traverse the list (L) and update the cluster label of each base station based on the following utility function
 - where

$$value(v, C) = con(v, C) \times \log\left(\frac{\tau}{\max\{dist(v, v')\}}\right)$$

 - v is the base station in consideration
 - C is the cluster that we suppose to integrate node v
 - $con(v, C)$ is the sum of the costs of the links in cluster C including node v
 - τ is the distance threshold between two nodes
 - $dist(v, v')$ is the distance between node v and node v' in cluster C
 - The base station is assigned to the cluster having the highest utility value
- Stop condition
 - Either the program reaches the maximum number of iterations
 - Or none of the BSs moved among clusters



Dataset

- Telecom Italia (TIM) Big Data Challenge
 - 2 months of traffic in Milan, Italie
- Base station positions
 - www.cellmapper.net

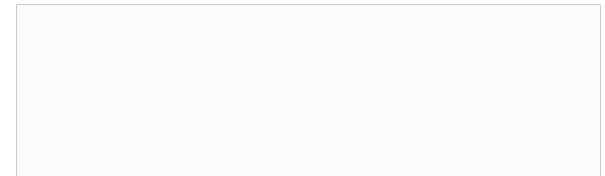
DATASET DESCRIPTION

Dataset	Item	Value
Network Traffic	# Grids	10,000
	Grid size	55,225 m^2
	Average traffic intensity	0.19
Base Station	# Base stations	182
	Average coverage	885,724 m^2
Data collection period		11/01/2013–12/31/2013

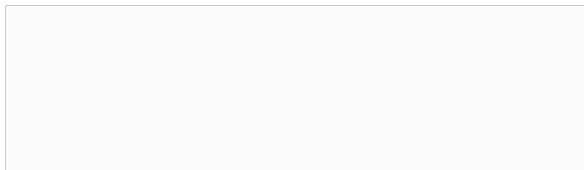
Performance metrics

- Gain in term of required network capacity

$$Cost(\mathbb{P}) = \frac{\sum_{C_k} \max \mathbf{f}(S_i)}{\sum_{s_i} \max \mathbf{f}(s_i)}$$



- $\mathbf{f}(S_i)$ is the profile of aggregated traffic of a base station in a cluster
- Gain in term of resource utilization



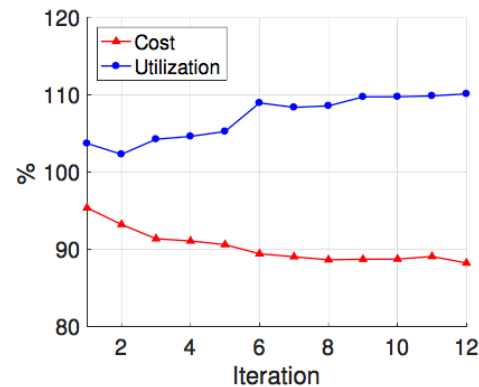
$$Util(\mathbb{P}) = \frac{\text{mean}_{C_k} U(C_k)}{\text{mean}_{s_i} U(s_i)}$$

- $U(C_k)$ is the average utilization rate of cluster C_k

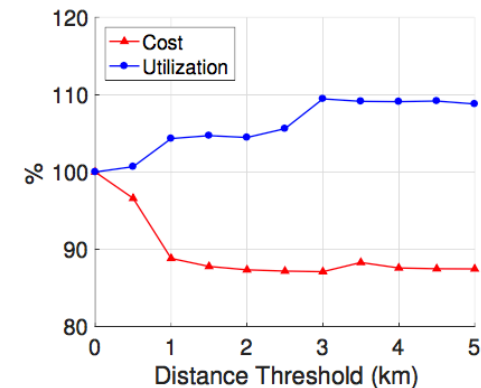
Performance analysis

- DCCA
 - Distance and Complementarity Constraint Algorithm
- DC
 - Distance constraint

	<i>Cost</i>	<i>Utilization</i>
Proposed (DCCA)	87.12%	109.45%
Baseline (DC)	96.68%	102.95%



(a)



(b)