Chuẩn bị dữ liệu với PANDAS

Người ta thường nói rằng 80% phân tích dữ liệu được sử dụng cho việc dọn dẹp và chuẩn bị dữ liệu. Chương này tập trung vào khía cạnh quan trọng là thao tác dữ liệu và làm sạch dữ liệu với Pandas.

pandas có hai kiểu dữ liệu chủ chốt

- Series: kiểu dữ liệu một chiều (1D) - chuỗi các giá trị
- DataFrame: kiểu dữ liệu hai chiều (2D) - tức sẽ biểu diễn thành bảng , có hàng / cột trên Excel

1. Series

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, Python objects, etc.). The axis labels are index.

```
# Nhập các thư viện cần thiết
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.stats as stats

# Tạo 1 series có 7 phần tử (chỉ mục từ 0) với các kiểu dữ liệu khác nhau
m = pd.Series([1,3,5,'c',6,8,"chuoi aa"])
print(m)
```

```
0         1
1         3
2         5
3         c
4         6
5         8
6    chuoi aa
dtype: object
```

2. DataFrame

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types.

```
columns = ['name', 'age', 'gender', 'job']
user1 = pd.DataFrame([['alice', 19, "F", "student"], ['john', 26, "M", "student"]],
columns=columns)
user2 = pd.DataFrame([['eric', 22, "M", "student"], ['paul', 58, "F", "manager"]],
columns=columns)
user3 = pd.DataFrame(dict(name=['peter', 'julie'], age=[33, 44], gender=['M', 'F'],
job=['engineer', 'scientist']))
print(user1)
print(user2)
print(user3)
```

```
   name  age gender     job
0  alice   19      F  student
1   john   26      M  student
  name  age gender     job
0  eric   22      M  student
1  paul   58      F  manager
   age gender       job   name
0   33      M   engineer  peter
1   44      F  scientist  julie
```

3. Kết nối DF

```
a = user1.append(user2)
users = pd.concat([user1, user2, user3])
print(a)
print("-----")
print(users)
```

```
    name  age gender      job
0  alice   19      F  student
1   john   26      M  student
0   eric   22      M  student
1   paul   58      F  manager
-----
    age gender        job   name
0   19      F    student  alice
1   26      M    student   john
0   22      M    student   eric
1   58      F    manager   paul
0   33      M   engineer  peter
1   44      F  scientist  julie
```

```
user4 = pd.DataFrame(dict(height=[165, 180, 175, 171],
       name=['alice', 'john', 'eric', 'julie'] ))
print(user4)
```

```
  height   name
0     165  alice
1     180   john
2     175   eric
3     171  julie
```

```
# Su dung cot name de ghep 2 DF.  Giao giữa các tên
merge_inter = pd.merge(users, user4, on="name")
print(merge_inter)
```

```
   age gender        job   name  height
0   19      F    student  alice     165
1   26      M    student   john     180
2   22      M    student   eric     175
3   44      F  scientist  julie     171
```

```
# Su dung cot name de ghep 2 DF. Hợp giữa các tên
print(users)
print("-----")
users = pd.merge(users, user4, on="name", how='outer')
print(users)
```

```
  age gender        job   name
0   19      F    student  alice
1   26      M    student   john
0   22      M    student   eric
1   58      F    manager   paul
0   33      M   engineer  peter
1   44      F  scientist  julie
-----
    age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
```

```
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
```

4. Examine data frame

users # print the first 30 and last 30 rows
type(users) # DataFrame
users.head() # print the first 5 rows
users.tail() # print the last 5 rows
print(users.describe()) # summarize all numeric columns
users.index # "the index" (aka "the labels")
users.columns # column names (which is "an index")
users.dtypes # data types of each column
users.shape # number of rows and columns
users.values # underlying numpy array
users.info() # summary (includes memory usage as of pandas 0.15.0)

```
             age      height
count   6.000000    4.000000
mean   33.666667  172.750000
std    14.895189    6.344289
min    19.000000  165.000000
25%    23.000000  169.500000
50%    29.500000  173.000000
75%    41.250000  176.250000
max    58.000000  180.000000
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6 entries, 0 to 5
Data columns (total 5 columns):
age       6 non-null int64
gender    6 non-null object
job       6 non-null object
name      6 non-null object
height    4 non-null float64
dtypes: float64(1), int64(1), object(3)
memory usage: 216.0+ bytes
```

5. Truy cập dòng

users['gender']   # select one column
type(users['gender']) # Series

```
users.gender          # select one column using the DataFrame

# select multiple columns
users[['age', 'gender']]    # select two columns
my_cols = ['age', 'gender'] # or, create a list...
users[my_cols]              # ...and use that list to select columns
type(users[my_cols])        # DataFrame
```

```
pandas.core.frame.DataFrame
```

6. Truy cập cột

```
# iloc is strictly integer position based
df = users.copy()
print(df)
df.iloc[0]     # first row
df.iloc[0, 0]  # first item of first row

df.iloc[0, 0] = 55

for i in range(users.shape[0]):
    row = df.iloc[i]
    row.age *= 100 # setting a copy, and not the original frame data.
print(df) # df is not modified
```

```
  age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
    age gender        job   name  height
0   55      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
C:\Users\lhtam\AppData\Local\Programs\Python\Python36-32\Lib\site-
packages\pandas\core\generic.py:3643: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

```
# ix supports mixed integer and label based access

df = users.copy()
df.ix[0]              # first row
df.ix[0, "age"]       # first item of first row
df.ix[0, "age"] = 55
for i in range(df.shape[0]):
    df.ix[i, "age"] *= 10
print(df)             # df is modified
```

```
  age gender        job   name  height
0  550      F    student  alice   165.0
1  260      M    student   john   180.0
2  220      M    student   eric   175.0
3  580      F    manager   paul     NaN
4  330      M   engineer  peter     NaN
5  440      F  scientist  julie   171.0
```

7. Bộ loc

```
# simple logical filtering
print(users[users.age < 20]) # only show users with age < 20
young_bool = users.age < 20 # or, create a Series of booleans...
young = users[young_bool] # ...and use that Series to filter rows
print(users[users.age < 20].job) # select one column from the filtered results
print(young)
```

```
  age gender      job   name  height
0   19      F  student  alice   165.0
0    student
Name: job, dtype: object
   age gender      job   name  height
0   19      F  student  alice   165.0
```

```
# Advanced logical filtering

users[users.age < 20][['age', 'job']]         # select multiple columns
users[(users.age > 20) & (users.gender=='M')] # use multiple conditions
print(users[users.job.isin(['student', 'engineer'])]) # filter specific values
```

```
  age gender       job   name  height
0   19      F   student  alice   165.0
1   26      M   student   john   180.0
2   22      M   student   eric   175.0
```

```
4   33      M  engineer  peter     NaN
```

8. Sorting

df = users.copy()
df.age.sort_values() # only works for a Series
df.sort_values(by='age') # sort rows by a specific column
df.sort_values(by='age', ascending=False) # use descending order instead
df.sort_values(by=['job', 'age']) # sort by multiple columns
df.sort_values(by=['job', 'age'], inplace=True) # modify df
print(df)

```
  age gender        job   name  height
4   33      M   engineer  peter     NaN
3   58      F    manager   paul     NaN
5   44      F  scientist  julie   171.0
0   19      F    student  alice   165.0
2   22      M    student   eric   175.0
1   26      M    student   john   180.0
```

9. Định hình lại

staked = pd.melt(users, id_vars="name", var_name="variable", value_name="value")
print(staked)

```
    name variable      value
0   alice      age         19
1    john      age         26
2    eric      age         22
3    paul      age         58
4   peter      age         33
5   julie      age         44
6   alice   gender          F
7    john   gender          M
8    eric   gender          M
9    paul   gender          F
10  peter   gender          M
11  julie   gender          F
12  alice      job    student
13   john      job    student
14   eric      job    student
15   paul      job    manager
16  peter      job   engineer
17  julie      job  scientist
18  alice   height        165
19   john   height        180
```

```
20   eric   height        175
21   paul   height        NaN
22  peter   height        NaN
23  julie   height        171
```

10. Xóa trùng lặp

df = users.append(df.iloc[0], ignore_index=True) # Gan 1 doi tuong vao 1 doi tuong da co
print(df)
#print(df.duplicated()) # Series of booleans. (True if a row is identical to a previous row)

print(df.duplicated().sum()) # count of duplicates
print(df[df.duplicated()]) # only show duplicates
print(df.age.duplicated()) # check a single column for duplicates -> Series
print(df.duplicated(['age', 'gender']).sum()) # specify columns for finding duplicates -> So dong trung lap theo cot
df = df.drop_duplicates() # drop duplicate rows
print(df)

```
  age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
6   33      M   engineer  peter     NaN
1
   age gender       job   name  height
6   33      M  engineer  peter     NaN
0    False
1    False
2    False
3    False
4    False
5    False
6     True
Name: age, dtype: bool
1
   age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
```

11. Dữ liệu thiếu

Missing values are often just excluded - loai

df = users.copy()
df.describe(include='all') # excludes missing values
print(df.describe(include='all'))

find missing values in a Series
print(df.height.isnull()) # True if NaN, False otherwise -> Series
print(df.height.notnull()) # False if NaN, True otherwise -> Series
print(df[df.height.notnull()]) # only show rows where height is not NaN
print(df.height.isnull().sum()) # count the missing values

find missing values in a DataFrame
print(df.isnull()) # DataFrame of booleans
print(df.isnull().sum()) # calculate the sum of each column bi thieu

```
          age gender      job  name      height
count    6.000000      6        6     6    4.000000
unique        NaN      2        4     6         NaN
top           NaN      F  student  john         NaN
freq          NaN      3        3     1         NaN
mean    33.666667    NaN      NaN   NaN  172.750000
std     14.895189    NaN      NaN   NaN    6.344289
min     19.000000    NaN      NaN   NaN  165.000000
25%     23.000000    NaN      NaN   NaN  169.500000
50%     29.500000    NaN      NaN   NaN  173.000000
75%     41.250000    NaN      NaN   NaN  176.250000
max     58.000000    NaN      NaN   NaN  180.000000
0    False
1    False
2    False
3     True
4     True
5    False
Name: height, dtype: bool
0     True
1     True
2     True
3    False
4    False
5     True
Name: height, dtype: bool
   age gender        job   name  height
0   19      F    student  alice   165.0
```

```
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
5   44      F  scientist  julie   171.0
2
      age  gender    job   name  height
0  False   False  False  False   False
1  False   False  False  False   False
2  False   False  False  False   False
3  False   False  False  False    True
4  False   False  False  False    True
5  False   False  False  False   False
age       0
gender    0
job       0
name      0
height    2
dtype: int64
```

a. Strategy 1: drop missing values

print(df)

m = df.dropna() # drop a row if ANY values are missing
print(m)

n = df.dropna(how='all') # drop a row only if ALL values are missing
print(n)

```
 age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
    age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
5   44      F  scientist  julie   171.0
    age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
```

b. Strategy 2: fill in missing values

```
df.height.mean()
df = users.copy()
df.ix[df.height.isnull(), "height"] = df["height"].mean()
print(df)
```

```
  age gender        job   name  height
0   19      F    student  alice  165.00
1   26      M    student   john  180.00
2   22      M    student   eric  175.00
3   58      F    manager   paul  172.75
4   33      M   engineer  peter  172.75
5   44      F  scientist  julie  171.00
```

12. Thay đổi giá trị

```
df = users.copy()
print(df.columns)
print(df)
print("----")

# Sưa tên cot "job"
df.columns = ['age', 'genre', 'travail', 'nom', 'taille']

# Định nghĩa thuộc tính travail va sua gia tri
df.travail = df.travail.map({ 'student':'etudiant', 'manager':'manager',
                              'engineer':'ingenieur', 'scientist':'scientific'})
print(df)
```

```
Index(['age', 'gender', 'job', 'name', 'height'], dtype='object')
   age gender        job   name  height
0   19      F    student  alice   165.0
1   26      M    student   john   180.0
2   22      M    student   eric   175.0
3   58      F    manager   paul     NaN
4   33      M   engineer  peter     NaN
5   44      F  scientist  julie   171.0
----
   age genre     travail    nom  taille
0   19     F    etudiant  alice   165.0
1   26     M    etudiant   john   180.0
2   22     M    etudiant   eric   175.0
3   58     F     manager   paul     NaN
```

```
4   33     M   ingenieur  peter     NaN
5   44     F  scientific  julie   171.0
```

13. Sử lý giá trị ngoại lai

- Based on parametric statistics: use the mean. Thay các giá trị ngoại lai bằng mean.

```
size = pd.Series(np.random.normal(loc=175, size=20, scale=10))
# Corrupt the first 3 measures
print(size)
print("____")
size[:3] += 500
print(size)
size_outlr_mean = size.copy() # Copy du lieu vao

# Duyet cac phan tu của X, neu (x-m) > 3* sd, thi thay bang mean.
size_outlr_mean[((size - size.mean()).abs() > 3 * size.std())] = size.mean()
print(size_outlr_mean.mean())
```

```
0     164.337714
1     185.508029
2     175.120042
3     172.421118
4     171.099672
5     179.902280
6     164.879989
7     167.406431
8     168.087546
9     180.422315
10    173.634678
11    161.940786
12    167.788481
13    197.850624
14    176.886892
15    176.030867
16    179.344483
17    194.504741
18    178.180654
19    182.311025
dtype: float64
____
```

```
0     664.337714
1     685.508029
2     675.120042
3     172.421118
4     171.099672
5     179.902280
6     164.879989
7     167.406431
8     168.087546
9     180.422315
10    173.634678
11    161.940786
12    167.788481
13    197.850624
14    176.886892
15    176.030867
16    179.344483
17    194.504741
18    178.180654
19    182.311025
dtype: float64
250.882918427
```

- Based on non-parametric statistics: use the median

```
# Median absolute deviation (MAD), based on the median, is a robust non-parametric statistics
mad = 1.4826 * np.median(np.abs(size - size.median()))
print(mad)
size_outlr_mad = size.copy()
size_outlr_mad[((size - size.median()).abs() > 3 * mad)] = size.median()
print(size_outlr_mad.mean(), size_outlr_mad.median())
```

```
11.7720872165
176.264695172 177.2103328050251
```

14. Groupby

```
for grp, data in users.groupby("job"):
    print(grp, data)
```

```
engineer    age gender       job   name  height
4   33      M  engineer  peter     NaN
manager    age gender      job  name  height
3   58      F  manager  paul     NaN
scientist    age gender        job   name  height
```

```
5   44      F  scientist  julie   171.0
student    age gender      job   name  height
0   19      F  student  alice   165.0
1   26      M  student   john   180.0
2   22      M  student   eric   175.0
```

15. File I/O

- Đọc và ghi dữ liệu từ Excel file

```
xls_filename = os.path.join(tmpdir, "users.xlsx")
users.to_excel(xls_filename, sheet_name='users', index=False)
pd.read_excel(xls_filename, sheetname='users')

# Multiple sheets
with pd.ExcelWriter(xls_filename) as writer:
users.to_excel(writer, sheet_name='users', index=False)
df.to_excel(writer, sheet_name='salary', index=False)
pd.read_excel(xls_filename, sheetname='users')
pd.read_excel(xls_filename, sheetname='salary')
```

16. Exercises

Data Frame

1. Read the iris dataset at 'https://raw.github.com/neurospin/pystatsml/master/data/iris.csv'
2. Print column names
3. Get numerical columns
4. For each species compute the mean of numerical columns and store it in a `stats` table like:

```
      species  sepal_length  sepal_width  petal_length  petal_width
0      setosa         5.006        3.428         1.462        0.246
1  versicolor         5.936        2.770         4.260        1.326
2   virginica         6.588        2.974         5.552        2.026
```

Missing data

Add some missing data to the previous table `users`:

```
df = users.copy()
df.ix[[0, 2], "age"] = None
df.ix[[1, 3], "gender"] = None
```

1. Write a function `fillmissing_with_mean(df)` that fill all missing value of numerical column with the mean of the current columns.
2. Save the original users and "imputed" frame in a single excel file "users.xlsx" with 2 sheets: original, imputed.