Data processing to data analysis

Using python.

Iván Andrés Trujilllo Abella

Facultad de Ingenieria
Pontificia Universidad Javeriana

trujilloiv@javeriana.edu.co

We need store the information!

We could need store and manipulate information, we can uses the built-in structure **list**

```
list = [0, 1.0, True, 'False']
```

Notice, that list allow us store information of different nature, floats, integers, stings and booleans.

Numpy and np.array()

np.array (ndarray) is the core of the **Numpy** library and its is array that allow us compute mathematical operations faster than list because of store elements of the same nature(type) (this allow us save resources as memory).

Pandas

It is a Open Source library that was written to facilitate the task of work with data. **provide** us two readable estructures:

Gives us **Series** and **DataFrames**

it is important remark that a Serie is similar to a ndarray (but the former include a index(or id)), this property is useful to change index values (for instance allow uses dates as indices).

Series

It is **One-Dimensional(1D)** structure composed of **key,value** in this case the **key** is named **index**:

- equal data type
- inmutable in size

0	'strA'
1	'strB'
2	'StrC'

dtype = object

Dataframes

Dataframe

is a collection of series....

	var1	var2	var2
0	'strA'	1.0	True
1	'strB'	1.1	False
2	'StrC'	1.2	False
dtype	Object	Int	Bool

It is **Two-Dimensional(2D)**

- not equal data type
- size mutable

implementation

```python
import numpy as np
import pandas as pd
pd.Series([10, 100, 1000], dtype=np.int64, name='tens')
type(tens.values) # The type of values are ndarray!
```

```python
income = pd.Series([11,10,22])
age = pd.Series([22,34,43])
df = pd.DataFrame({'income':income, 'age':age})
display(df)
#Notice the following:
```

series in pandas

A arrangement composed of data the same types and index.

```
import pandas as pd
pd.Series().reset_index(drop=True)
pd.Series().reset_index()
```

operations over series

```
get(index)
loc[index_start: index_end] # include last
iloc[position_start: position_end] # do not include last
```

Missing values in Pandas

Pandas is related with **Numpy**,

```
n = float('nan')
```

This value not is compare with itself to check if is a missing value you can uses $math.isnan(n)$.
the $==$ operator does novt produce the expected result.

Missing values in rows and columns

```
import pandas as pd
import numpy as np
df = pd.DataFrame([[’line’,float(’nan’),float(’nan’)] ,
    [’circle’,’red’,10], [float(’nan’),’black’,1]],
    columns=[’figure’,’color’,’value’])
print(df)
df[’figure’].isna().sum()
df.loc[0].isna().sum()
```

dropna method

A dataframe method,

```
n = float('nan')
```

replace a value

```
df.replace('nan',float('nan'), inplace=True)
```

add

hello

Special topics in python

lambda expression, map, reduce, filter, zip and lambda are topics very usual and a bit confusing to a novel.

lambda expression

Until now we have been using the keyword: note that using lambda we not specify a name, lambda expressions are also called **anonymous functions**.

```
def f(x,y):
    return x**y
```

```
f  = lambda x,y : x**y
f(10,2)
```

map

map is used to apply a function to each element in a iterable object.

replace

```
string = ’latex’
string.replace(’t’,’T’)
```

split and join

```
joined =’-’.join((’a’,’b’,’c’))
joined.split("-")
```

Regular expression

Regexs are very useful to handle strings and perfoming operations over them.

regular expressions

```
import re
re.search(’d[a-z]f’,’ainingdaf’)
```

this will return the same result as

```
re.search(’d[a-z]f’,’ainingdef’)
```

Searching words

```
hello
```

excluding characters

```
''.join([i for i in s if not i.isdigit()])
```

```
''.join(filter(lambda x: not x.isdigit(), mystr))
```

multiple replacement

```
s = ’one two one two one’

print(s.translate(str.maketrans({’o’: ’O’, ’t’: ’T’})))
# One TwO One TwO One

print(s.translate(str.maketrans({’o’: ’XXX’, ’t’: None})))
# XXXne wXXX XXXne wXXX XXXne
```

re.sub()

so far, we have seen that the old string must match completely, however we can uses re.sub() and re.subn()

```
re.sub(’regex’,’ new string’ , ’string to change’)
```

group by

is as a bysort in stata. this section require attention and it is a powerful techniques to work with data.

Merge

we have two datasets dataset a and dataset b, anad keys in key a and key b.

```
merge = pd.merge(dataA, dataB, left_on='key A', right_on='Key B')
```

```
merge = pd.merge(dataA, dataB, left_on='key A', right_on='Key
    B', indicator=True)
```

the main difference with join is that join is based in the index. What it is the value of **how=** by default?

outer

id	city	pop
'1.1'	A	10
'2.1'	B	11
'3.1'	C	12
'4.1'	D	13

id	guns
'2.1'	'legal'
'3.2'	'ilgeal'
'1.1'	'legal'
'4.2'	'legal'

```
merge = pd.merge(df1,df2, on='id', indicator=True, how='outer')
```

id	city	pop	guns	_merge
1.1	A	10.2	legal	both
2.1	B	11.0	legal	both
3.1	C	4.0	NaN	left_only
4.1	D	5.0	NaN	left_only
3.2	NaN	NaN	ilegal	right_only
4.2	NaN	NaN	legal	right_only

Possible problems

```
df_one = {
'id':
['colombia','colombia','colombia','zimbawe','zimbawe','argentina'],
'year':
[2000,2001,2002,2000,2001,2003],
'pop':
[100,120,311,220,110,331]
}
df_two={
'id':
['zimbawe','zimbawe','argentina','colombia','colombia','colombia'],
'year':
[2001,2000,2014,2001,2002,2003],
'gdp':
[10,11,31,22,14,6]
}
```

take in mind that **on** could be a list if there are more of one identifier as in panel data.

```
df_one = pd.DataFrame(df_one)
df_two = pd.DataFrame(df_two)
print(df_one[’id’].duplicated())

merge_df = pd.merge(df_one, df_two, on=[’id’,’year’],
    how=’outer’, indicator=True)
```

df[’var’].unique()

```
import pandas as pd
df =
    pd.DataFrame({’cole’:[’a’,’a’,’b’,’b’,’c’,’c’,’c’,’d’,’e’,’e’],
    ’point’:[1,2,3,4,5,6,7,8,9,10]})
df[’cole’].unique()
```

return the unique values 'a','b','c','d','e'.

how is used to model data np.where()?

pd.get$_{d}ummies(data[var], prefix = var)$

Creating aggregate variables

```
bysort drug: egen medp = mean(studytime)
```

```
gb = df.groupby(’drug’)[’studytime’]
df[’means’] = gb.transform("mean")
df
```

filtering

```
import pandas as pd
df =
    pd.DataFrame({’cole’:[’a’,’a’,’b’,’b’,’c’,’c’,’c’,’d’,’e’,’e’],
    ’point’:[1,2,3,4,5,6,7,8,9,10]})
print(df)
df[df[’cole’].isin(["a", "b", "d"])]
```

Quantil

```
import numpy as np
def quantileArray(array): ## A Quantile array
  a =list(np.quantile(array,[0.25,0.50,0.75]))
  arrayq = np.where(array<=a[0], ’Q1’, np.where(array<=a[1],
      ’Q2’, ’Q3’))
  return list(arrayq)
```

To groups

```
import pandas as pd
programa  = [
’economics’,’economics’,’economics’,’economics’,
’economics’,’law’,’law’,’law’
]

notas_lectura = [10,9,8,4,1,10,4,3]

notas_math = [11,23,44,55,66,21,12,11]

df = pd.DataFrame({’Programa’:programa,
    ’notas_lectura’:notas_lectura , ’notas_math’:notas_math} )

df.groupby(’Programa’).agg( mean = (’notas_math’,
    np.sum),quantiles = (’notas_lectura’, lambda x:
    quantileArray(x)))
```

To groups

equal size

```
df['mean'] =
    df.groupby('Programa')['notas_lectura'].transform('mean')
df['Quantiles'] =
    df.groupby('Programa')['notas_lectura'].transform(lambda x:
    quantileArray(x))
```

loc and iloc

subset

```
data.iloc['rows', 'columns']
```

iloc is based in index or position while loc is based in labels or conditionals. if we ommited the second parameter then, will be selected all colummns in the specific row. Otherwise to slecet a column you need put :

```
data.iloc[:,-1]
```

this will select the last column with all rows.

```
df.iloc[:,[0,2,1]] $ first, third and second column
```

loc and conditions

```
is_male = df.loc[:, ’sex’] == ’Male’
df_male = df.loc[is_male]
df_male.head()
```

take in mind that will be return series or dataframe.

Preprocessing

```
df[’ageds’] =
    [(df.loc[x,’age’]-np.mean(df[’age’]))/np.std(df[’age’]) for
    x in df.index]
```

```python
from sklearn.preprocessing import scale
scale(X)
```

Reshaping and Pivot Tables

Retention rate

Cohort	users	period 0 day(0)	period 1 day(1)	period 2	Period 3
1-febr	10	100%	40%	33%	12%
2-febr	22	99.%	98%	.	.
3-febr	34	98.4%	.	.	.

The description of the table coul be, **users** indicate the number of clients that ares inscrited in **1-feb** and since how many days contiue in the application, for instance to the another day 40% ramain in the application.