1ª Emisión

Módulo 05

Manipulación y visualización de datos con Python

Mtro. Ricardo Daniel Alanis Tamez

DGTIC

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC

Educación Continua
1971 - 2021

Agrupación y Agregación de Datos

Manipulación y visualización de datos con Python

Ricardo Alanís

Presentación

En este tema tendremos la oportunidad de ver las diferentes rutas para transformar y agregar información.

Objetivo

El participante identificará los diferentes métodos de agregación y agrupación de datos.

Agenda de hoy

1. **Paseo por las agregaciones disponibles**
2. **Exposición de alternativas para el trabajo de datos**

Cheatshe

Categorizing a data set and applying a function to each group, whether an aggregation or transformation.

Note	Aggregation of "Time Series" data - please see Time Series section. Special use case of "groupby" is used - called "resampling".

GROUPBY (SPLIT-APPLY-COMBINE)

- Similar to **SQL groupby**

Compute Group Mean	`df1.groupby('col2').mean()`
GroupBy More Than One Key	`df1.groupby([df1['col2'], df1['col3']]).mean()` # result in hierarchical index consisting of unique pairs of keys
"GroupBy" Object : (ONLY computed intermediate data about the group key - `df1['col2']`	`grouped = df1['col1'].groupby(df1['col2'])` `grouped.mean()` # gets the mean of each group formed by 'col2'

Indexing "GroupBy" Object	# select 'col1' for aggregation :
	`df1.groupby('col2')['col1']`
	or
	`df1['col1'].groupby(df1['col2'])`

Note	Any missing values in the group are excluded from the result.

1. Iterating over GroupBy object

"GroupBy" object supports iteration : generating a sequence of 2-tuples containing the group name along with the chunk of data.

```
for name, groupdata in df1.groupby('col2'):
```
name is single value, groupdata is filtered DF contains data only match that single value.

```
for (k1, k2), groupdata in df1.
groupby(['col2', 'col3']):
```
If groupby multiple keys : first element in the tuple is a tuple of key values.

Convert Groups to Dict	`dict(list(df1.groupby('col2')))` # col2 unique values will be keys of dict

Group Columns by "dtype"	`grouped = df1.groupby([df1.dtypes, axis = 1)`
	`dict(list(grouped))`
	# separates data into different types

DATA AGGREGATION

Data aggregation means any data transformation that produces **scalar** values from arrays, such as "mean", "max", etc.

Use Self-Defined Function	`def func1(array): ...` `grouped.agg(func1)`
Get DF with Column Names as Fuction Names	`grouped.agg([mean, std])`
Get DF with Self-Defined Column Names	`grouped.agg([('col1', mean), ('col2', std)])`
Use Different Fuction Depending on the Column	`grouped.agg({'col1' : [min, max], 'col3' : sum})`

GROUP-WISE OPERATIONS AND TRANSFORMATIONS

Agg() is a special case of data transformation, aka reduce a **one-dimensional array to scalar.**

Transform() is a specialized data transformation :

- It applies a function to each group, if it produces a scalar value, the value will be placed in **every row** of the group. Thus, if DF has 10 rows, after "transform()", there will be still 10 rows, each one with the scalar value from its respective group's value from the function.
- The passed function must either produce a scalar value or a transformed array of same size.

General purpose transformation : apply()

```
df1.groupby('col2').apply(your_func1)
```
your func ONLY need to return a pandas object or a scalar.

Example 1 : Yearly Correlations with SPX

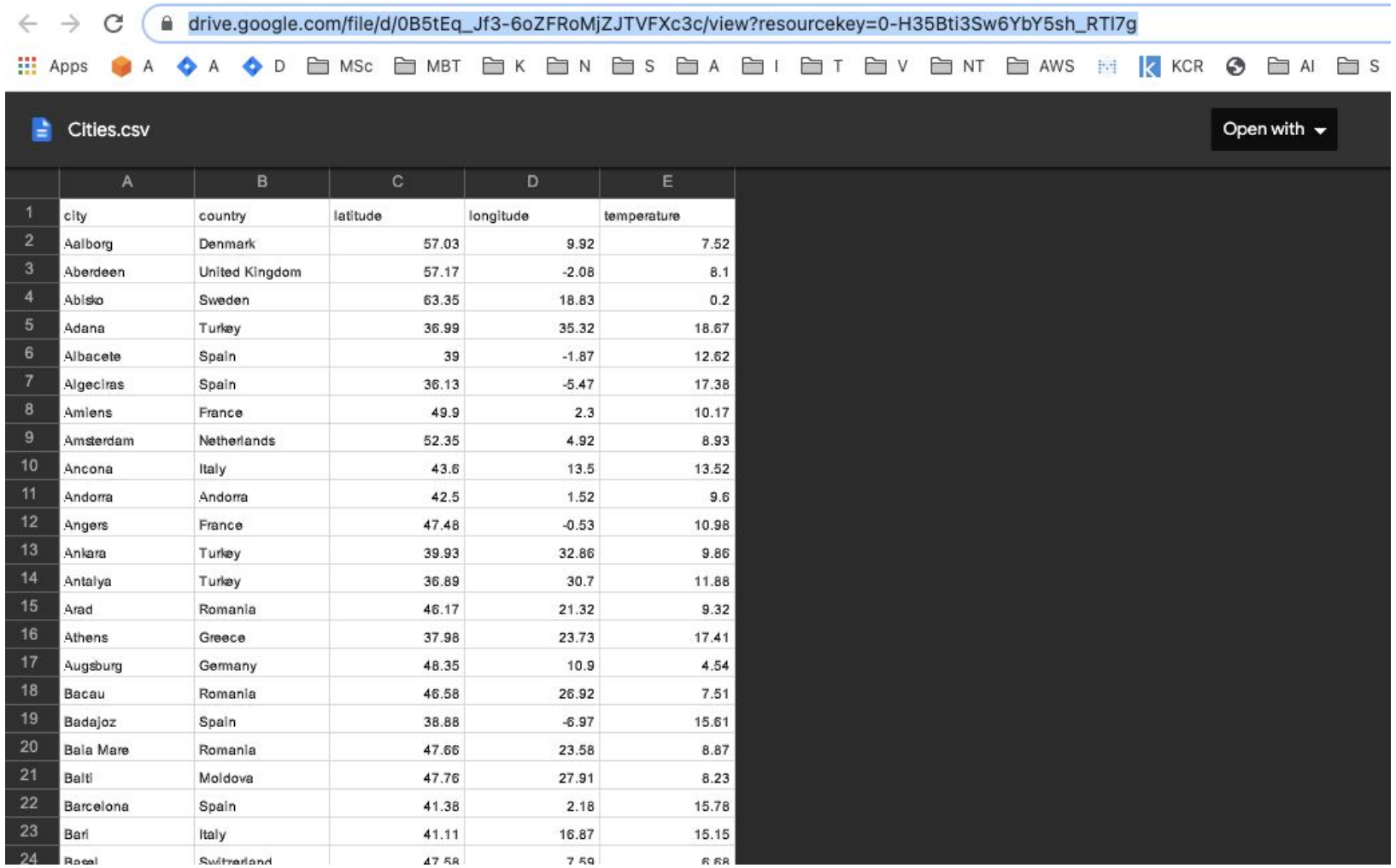
"close_price" is DF with stocks and SPX closed price columns and dates index

```
returns = close_price.pct_change().dropna()
by_year = returns.groupby(lambda x :
x.year)
spx_corr = lambda x : x.corrwith(x['SPX'])
by_year.apply(spx_corr)
```

Example 2 : Exploratory Regression

```
import statsmodels.api as sm
def regress(data, y, x):
    Y = data[y]; X = data[x]
    X['intercept'] = 1
```

Usemos un dataset de referencia



	A	B	C	D	E
1	city	country	latitude	longitude	temperature
2	Aalborg	Denmark	57.03	9.92	7.52
3	Aberdeen	United Kingdom	57.17	-2.08	8.1
4	Abisko	Sweden	63.35	18.83	0.2
5	Adana	Turkey	36.99	35.32	18.67
6	Albacete	Spain	39	-1.87	12.62
7	Algeciras	Spain	36.13	-5.47	17.38
8	Amiens	France	49.9	2.3	10.17
9	Amsterdam	Netherlands	52.35	4.92	8.93
10	Ancona	Italy	43.6	13.5	13.52
11	Andorra	Andorra	42.5	1.52	9.6
12	Angers	France	47.48	-0.53	10.98
13	Ankara	Turkey	39.93	32.86	9.86
14	Antalya	Turkey	36.89	30.7	11.88
15	Arad	Romania	46.17	21.32	9.32
16	Athens	Greece	37.98	23.73	17.41
17	Augsburg	Germany	48.35	10.9	4.54
18	Bacau	Romania	46.58	26.92	7.51
19	Badajoz	Spain	38.88	-6.97	15.61
20	Bala Mare	Romania	47.66	23.58	8.87
21	Balti	Moldova	47.76	27.91	8.23
22	Barcelona	Spain	41.38	2.18	15.78
23	Bari	Italy	41.11	16.87	15.15
24	Basel	Switzerland	47.58	7.59	6.68

Groupby (split-apply-combine)

- Cómputo por funciones predefinidas (Ver **referencia**)

df.groupby('col').mean()

- Groupby de más de una llave

df.groupby(df[col1], df[col2]).mean()

SOLID PRINCIPLES

S — SINGLE RESPONSIBILITY → A class should have only single responsibility and should have one and only one reason for change

O — OPEN CLOSED PRINCIPLE → A class should be open for extension, but closed for modifications

L — LISKOV SUBSTITUTION → Objects in a program should be replaceable with instances of their subtypes without altering the correctness of program

I — INTERFACE SEGREGATION → Segregate Intefaces as per the requirments of program, rather than one general purpose Implementation

D — DEPENDENCY INVERSION → Should depend on abstractions rather than concrete implementations

@javatechonline.com

The Wrong Abstraction

I originally wrote the following for my Chainline Newsletter, but I continue to get twee about this idea, so I'm re-publishing the article here on my blog. This version has been lightly edited.

I've been thinking about the consequences of the "wrong abstraction." My RailsConf 20 "all the little things" talk included a section where I asserted:

> *duplication is far cheaper than the wrong abstraction*

And in the summary, I went on to advise:

Groupby (split-apply-combine)

- Usando una función sobre un objeto agrupado (obtiene la media de cada grupo formado por col2)

```
grouped = df[‘col1’].groupbydf[‘col2’])

grouped.mean()
```

- Indexando un objeto “Groupby”

```
df.groupby(‘col1’)[‘col2’]
```

Acceso a los datos

Si solo es un solo valor:

```
for name, groupdata in df1.groupby(‘col2;)
```

Si son dos valores:

```
for (k1,k2), groupdata in df1.groupby(‘col2’, ‘col3’):
```

También es posible convirtiendo a diccionarios o por dtypeÑ

```
dict(list(df1.groupby(‘col2’))
```

```
grouped = df1.groupby([df1.dtypes, axis=1)
dict(list(grouped)
```

Agrupando por funciones

Cuando se usa una función como llave de agregación, la función se aplicará a los grupos de respuesta similares.

Por ejemplo:
df1.groupby(len).sum()

Retornará un dataframe con el tamaño de índice de filas del tamaño de los nombres. Así, los nombres del mismo tamaño sumarán sus valores.

Agregación de Datos

Cualquier transformación que produce un escalar desde arreglos como min, max, etc. Puede ser:

Una función propia **grouped.agg(func1)**

Un arreglo de funciones **grouped.agg(mean, std)**

Un arreglo de columnas y funciones **grouped.agg([('col1', mean), ('col2', std)**

Un diccionario de columnas y funciones **grouped.agg({'col1': [min, max],...})**

Transformaciones en grupo y operaciones

.agg es una transformación especial, de forma que se forma un arreglo de una dimensión a escalar

Transform es una transformación especial, que aplica a cada fila y la función debe de generar un escalar.

df1.groupyby('col2').apply(your_func1)

¿Preguntas?

Referencias

- “pandas.DataFrame.aggregate.” pandas. pandas.pydata.org/docs/reference/api/pandas.DataFrame.aggregate.htm
- Shane, “Pandas Groupby: Summarising, Aggregating, Grouping in Python.” Shane Lynn. www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/
- “Aggregation and Grouping.” Python Data Science Handbook. jakevdp.github.io/PythonDataScienceHandbook/03.08-aggregation-and-grouping.html

Contacto

Mtro. Ricardo Daniel Alanis Tamez

ricardo@codeandomexico.org

LinkedIn: Ricardo Alanís