Hecho un dataframes, Importa y Exporta los datos

Andrew Reeve
School of Earth and Climate Sciences
University of Maine

Qué es Pandas (Panel Data)?

- Tipos de datos nuevos
  - Series: Una-dimensión con indices
  - DataFrame: multi-dimensión con indices
    - estructura como hoja de cálculo
- Añadir nombres a los filas y columnas
- asignación inteligente de NaN's
- matemáticas optimizado (eg. rolling averages, interpolation)
- entrada/salida de los datos de archivo
- graficas con matplotlib & seaborn

Series

- hecho con metido de `Series`
  - 1-D conjuntos de datos, como vector de numpy
  - puede dar indices (nombres de filas)
- hecho de:
  - lists de valores y opcional indices (también una lista)
  - diccionario (con clave para cada elemento)
- puede dar un nombre a una series
- puede combinar los series
- extrear datos de una series
  - `loc` (locación): usa nombre de indices
  - `iloc`: usa la posicional indices

```
import pandas as pd
from string import ascii_letters

a_list = [100+i for  i in range(10)]
b_list = [100+i for i in range(8,16)]

a_index=[ascii_letters[i] for i in range(10)]
b_dict={ascii_letters[i]:b_list[n] for n,i in enumerate(range(8,16))
        }

a_series=pd.Series(a_list, index=a_index)
b_series=pd.Series(b_dict)

a_series.name='a_name'
b_series.name='b_name'
# note how data handled when combining
add_series=a_series+b_series
concat_series=pd.concat((a_series,b_series))

# getting data from a series
print(a_series['a'])
# note that this included 'd'
print(a_series['a':'d'])
#location command, prefered way of slicing
print(a_series.loc['a':'d'])
# index location, using positional indices
# does not unclude '4'
print(a_series.iloc[0:4])
# get index
print(a_series.index)
# get values
print(a_series.values)
```

DataFrames

- metedo `DataFrame` (CamelCase)
  - 2-D (más posible)
  - filas y columnas con nombres
- hecho con diccionarios (y otras maneras)
  - clave es nombre de columna, listas tienen los valores
- añada columnas con `assign`
- cambiar indices
  - `set_index`: usa columna como idx & retira
  - `df.index=values`: sobrescribe idx
  - `df.drop([list if row/column names])`: retirar filas,columnas
    - predeterminado a filas, puede especificar axis

```
import pandas as pd
# some data
data_dict={'month':['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct
    ','Nov','Dec'],
           'Bgr_temp':[18,20,30,42,54,64,68,67,58,48,37,24],
           'Bgr_ppt':[3.0,2.9,3.1,3.3,3.6,3.3,3.3,3.4,3.4,3.4,4.6,4.0],
           'Blfst_temp':[21,24,34,44,54,64,68,67,58,48,38,26],
           'Blfst_ppt':[3.5,3.3,3.9,4.4,4.3,3.5,3.1,3.3,3.6,4.3,5.4,4.9]}
# dataframe from series
# only difference is that *a_df* has a column heading
a_series=pd.Series(data_dict['Bgr_temp'])
# specify index
a_series=pd.Series(data_dict['Bgr_temp'], index=data_dict['month'])
a_df=pd.DataFrame(a_series)
# specify index using dict of lists
a_df=pd.DataFrame({'Brg_temp':data_dict['Bgr_temp']},index=data_dict['month'])
b_df=pd.DataFrame({'Brg_temp':data_dict['Bgr_temp'],
        'Bgr_ppt':[3.0,2.9,3.1,3.3,3.6,3.3,3.3,3.4,3.4,3.4,4.6,4.0],},
       index=data_dict['month'])
# convert entire dict into dataframe
c_df=pd.DataFrame(data_dict)
# adding a column to df
c_df=c_df.assign(month2=c_df.month.str.lower())
# setting index using column, removes column from df
c_df = c_df.set_index('month')
# overwriting index values with another index
c_df.index = c_df['month2']
# dropping a column, need to indicate column axis, row (axis=0) is default
c_df.drop(['month2'],axis=1)
```

Leer y Escribir Archivos

- Muchos módulos de `read_xxx`:
  - Leer archivo a dataframe
  - e.g., read_{csv}, read_{hdf}, $\text{read}_{\text{table}}$
  - Algunas mandas tienen muchísimas opciones confusas de palabras de clave
- Módulos de `to_xxx`
  - Ahorrar dataframe a archivo
  - e.g., to_{hdf}, to_{csv}, to_{json}
- Debe limpiar los archivos antes de ahorrar
  - Se cae filas/columnas no necesita
  - Cambia tipos de datos de columnas (`astype`)
  - Cambia nombres de indice/columna (`set_index`, `index`, `column`)

```
import pandas as pd

# reading a csv, there are a confusingly large number of keywords for this
df1=pd.read_csv('./examples/CB_Yellow_Rod_Green_75.csv',
                skiprows=47, skipfooter=2, engine='python',
      parse_dates=['TimeStamp'],)
# saving dataframe to a file
df1.to_csv('./examples/ibutton_frame.csv.gz',compression='gzip')
df1.to_hdf('./examples/ibutton_frame.hdf','root', mode='w', complib='blosc')

# reading a stream gauging file, USGS table format (tab delimited)
df2=pd.read_table('./examples/USGS01034000.tab',comment='#')

# cleaning up the file
df2=df2.drop([0])
df2=df2.drop(['agency_cd', 'site_no', 'tz_cd', '65543_00060_cd', '65544_00065_cd'],
             axis=1)
df2=df2.set_index('datetime')
#check data types, they are strings! Need to convert
df2.loc[:,'65544_00065']=df2.loc[:,'65544_00065'].astype('float')
df2.loc[:,'65543_00060']=df2.loc[:,'65543_00060'].astype('float')
```

Combinar Dataframes

- diferentes mandas para combinar df: `concat`, `merge`, `join`
  - `concat` 'apila' los dataframes
  - `merge` combina usando columna en común entre dataframes
  - `join` combina usando indices
- puede fabrica una dataframe vació y añade columnas/filas
- `multi-index` para organizar los datos

```
import pandas as pd
import numpy as np
from string import ascii_uppercase,
    ascii_lowercase

## make some dataframes
list1 = list(ascii_uppercase)
list2 = list(ascii_lowercase)

df1 = pd.DataFrame({"upper": list1[0:12], "lower"
    : list2[0:12]})
df2 = pd.DataFrame({"upper": list1[9:26], "lower"
    : list2[9:26]},
        index=range(9, 26))

## concatenate frames
df3 = pd.concat((df1, df2))
df4 = pd.concat((df1, df2), axis=1)
```

Más Combina

```
import pandas as pd
import numpy as np
from string import ascii_uppercase,
    ascii_lowercase

## make some dataframes
list1 = list(ascii_uppercase)
list2 = list(ascii_lowercase)

df1 = pd.DataFrame({"upper": list1[0:12], "
    lower": list2[0:12]})
df2 = pd.DataFrame({"upper": list1[9:26], "
    lower": list2[9:26]},
        index=range(9, 26))

## concatenate frames
df3 = pd.concat((df1, df2))
df4 = pd.concat((df1, df2), axis=1)
```


```
import pandas as pd
import numpy as np
from string import ascii_uppercase,
    ascii_lowercase

## make some dataframes
list1 = list(ascii_uppercase)
list2 = list(ascii_lowercase)

df1 = pd.DataFrame({"upper": list1[0:12], "
    lower": list2[0:12]})
df2 = pd.DataFrame({"upper": list1[9:26], "
    lower": list2[9:26]},
        index=range(9, 26))

## concatenate frames
df3 = pd.concat((df1, df2))
df4 = pd.concat((df1, df2), axis=1)
```

Converting Datatypes and Null Values

- Los datos lean como cadeneas a pandas (e.g. archivos de csv)
- Convierte a int, float or datetime
  - `astype` convierte los tipos
  - `pandas.to_datetime(dataframe)` convierte a objectos de `datetime`
  - `pd.to_numeric(dataframe)` convierte a números
  - `errors` palabra de clave, para manejar errores (`coerce`, `ignore`, `raise`)
- probar NaNs
  - funciones de pandas para ID valores 'null'
    - `isna` o `isnull`, `notna` o `notnull`
  - `any` y `all` prueban para valores `True`
- `dropna` retira valores 'null'
- `fillna` llena valores de 'null' valores con otros valores

```
import pandas as pd
import numpy as np
# all columns read into dataframe as 'objects', an object datatype
      in numpy and pandas is a string with variable lengths
df = pd.read_csv("annual_peak_flow.txt", comment="#",
                 delimiter="\t")
# can see dtypes fromall columns using:
df.dtypes # with 's'
# or individual colum dtype with:
df.peak_dt.dtype # no 's'
# remove first row
df = df.drop([0,])
# remove uneeded columns
df = df.drop(["agency_cd", "peak_tm", "year_last_pk", "ag_dt",
              "ag_tm", "ag_gage_ht_cd","peak_cd", "gage_ht_cd",],
             axis=1,)
# raises an error, a date is '2001-00-00', makes no sense
# df.loc[:,'peak_dt']=pd.to_datetime(df.loc[:,'peak_dt'])

# ID problem dates and overwrite 'bad' data with NA values
date_bool = df.peak_dt.str[-2:] == "00"
df.loc[date_bool, "peak_dt"] = pd.NA
# convert column data type
df.loc[:, "peak_dt"] = pd.to_datetime(df.loc[:, "peak_dt"]).astype("
      datetime64[ns]")
# easier ? way of converting to datetime
# df.loc[:,'peak_dt']=pd.to_datetime(df.loc[:,'peak_dt'],errors='
      coerce')
df.loc[:, "peak_va"] = pd.to_numeric(df.loc[:, "peak_va"], errors="
      coerce")
```

Selecciona los datos de DataFrame

- para sacar una parte de dataframe
  - `df.loc[rows,cols]` seleccione por nombres de filas/col
  - `df.iloc[rows,cols]` seleccione por indices posicional
  - combina estos con operaciones booleanas
- mira rápida en DataFrame
  - `df.head(rows)` vuelve primeras filas
  - `df.tail(rows)` vuelve ultimas filas
  - `df.info()` info básico de DataFrame
  - `df.describe()` estadísticas básicas sobre columnas

```
import pandas as pd
import numpy as np

adict={'a':[1,2,3],'b':['x','y','z']}
df=pd.DataFrame(adict, index=['a0','a1','a2'])

# getting just values of 1 and 2 from dataframe
# method 1, slicing with loc
m1=df.loc[['a1','a2'], 'a']

# method 2, slicing with iloc
m2=df.iloc[[1,2], 0]

# method 3, using booleans

m3 = df.loc[[False,True,True],'a']

# method 4, using boolian expression
# these do the same thing
m4=df.loc[df.b!='x','a']
m4=df.loc[df.a>1,'a']
# replacing a value using boolean opperation
df2=df.copy()
df2.loc[df2.a>1,'a'] = df2.loc[df2.a>1,'a']+20
df2.loc[df2.a>1,'a'] = df2.loc[df2.a<=1,'a']+10
```